AD-A060 780   MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78
F19628-76-C-0197
UNCLASSIFIED   ESD-TR-78-130-VOL-2   NL

1 OF 6
AD
A060780

ESD-TR-78-130, Vol. II

A058629

USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL PART II (REFERENCE) H6180/MULTICS/VERSION 3.3

ISDOS Project
University of Michigan
Department of Industrial and Operations Engineering
Ann Arbor, Michigan 48109

July 1978

A058 629

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

78 10 23 165

This Technical Report has been reviewed and is approved for publication.

[illegible]
ANN MELANSON
Project Engineer

[illegible]
CHARLES J. GREWE, Jr., Lt/Col, USAF
Chief, Technology Applications Division

FOR THE COMMANDER

[illegible]
ERIC B. NELSON, Colonel, USAF
Acting Director, Computer Systems Engineering
Deputy for Technical Operations

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

(19) **REPORT DOCUMENTATION PAGE**	**READ INSTRUCTIONS BEFORE COMPLETING FORM**
1. REPORT NUMBER (18) ESD-TR-78-130- VOL- [redacted] 2 — 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER (9) Technical rept.
4. TITLE (and Subtitle) (6) USER REQUIREMENTS LANGUAGE (URL) User's Manual. Part II. (Reference) H6180/Multics/Version 3.3.	5. TYPE OF REPORT & PERIOD COVERED Jul[redacted]76-[redacted] Mar[redacted]77, [redacted]
	6. PERFORMING ORG. REPORT NUMBER CDRL Item 021
7. AUTHOR(s) ISDOS Project	8. CONTRACT OR GRANT NUMBER(s) (15) F19628-76-C-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Michigan Department of Industrial & Operations Engineering Ann Arbor, MI 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE64740F, Project 2237 (16)
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Technical Operations Electronic Systems Division Hanscom AFB, MA 01731	12. REPORT DATE (11) Jul[redacted]78
	13. NUMBER OF PAGES 447
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 504 p.	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Computer-Aided Design
Information Processing
Information System Requirements
Requirements Analysis
Requirements Language
Requirements Specification
Specification Analysis

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report is part of a series that deals with a Computer-Aided Design and Specification Analysis Tool (CADSAT). The purpose of the tool is to describe the requirements for information processing systems and to record such descriptions in machine-processable form. The major components of CADSAT are the User Requirements Language (URL) and the User Requirements Analyzer (URA) which can operate in an interactive computer environment. This report, Part I and Part II, describes how the formal URL may be used to define systems. It explains the language statements available, their use and application on a Honeywell 6180

over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 871

gm

20. ABSTRACT (Continued)

Multics Computer.

PREFACE

This manual describes the User Requirements Language (URL) to be used with Version 3.3 of the User Requirements Analyzer (URA). The manual consists of two volumes which are referred to as Part I and Part II in the documentation. Part I gives a detailed description of the URL statements available and their use. Part II is a reference manual which gives the proper syntax for each statement.

TABLE OF CONTENTS

LIST OF APPENDICES

1. INTRODUCTION AND PURPOSE

The original Problem Statement Language (PSL 1.0) was designed to provide the User with an improved method of stating requirements for a target information processing system (IPS). This goal was achieved by developmental work in the ISDOS Research Project leading to PSL 2.0 and URL 3.0 and their associated Analyzers (PSA 2.0 and URA 3.0). However, as with any developmental project, continued work yields improved understanding and eventually an improved product. Such is the case for URL 3.2 and the URA 3.2.

The new URL 3.2, hereafter referred to as URL, provides the User greater flexibility, more features and greater ease of use, while still maintaining the overall goals of such a computer-aided method. Therefore, URL is designed to provide understandable communication and documentation for both men and machine by having a simple syntax for the machine while maintaining the readability for the man.

The purpose of this manual is to provide a concise description of URL syntax and give brief examples of usage.

2. THE LANGUAGE

2.1 Introduction

Any language which is to be processed by computer needs to be structured in some way. The User Requirements Language, although it is based on English in that it uses English words and is intended to be readable as English text, must therefore be more precise than a natural language. Just as in English, the basic unit of the language is a word. In order for the Analyzer to understand URL, it treats all words as one of two types: Reserved Words, and names. Reserved Words have a specific meaning to the Analyzer and must be spelled exactly as given in the Reserved Word List (Appendix B). Many Reserved Words have a short form which may be substituted for the Reserved Word; these short forms are also given in the Reserved Word List. Some Reserved Words are essential for the URA to interpret the meaning of a statement. Other Reserved Words are not used by the Analyzer. These Reserved Words are called Optional Words (see Appendix C). Names are assigned by the User to facilitate the description of the target system. Names must be formed according to the rules given in sections 2.3 and 2.5.

These Reserved Words and names are combined with appropriate punctuation to form statements. Punctuation must be given exactly as shown in the syntax for a statement. For example, name(s) correspond to several names separated by commas; the commas are required in name(s) between each pair of names. A special punctuation symbol, a semi-colon, is used to end a statement in URL. Just as some Reserved Words are optional and do not affect the interpretation of a statement by the Analyzer, the colon is a special punctuation which may be used without affecting the meaning of a statement.

To illustrate, the syntax for the KEYWORD statement is:

  KEYWORDS ARE keyword-name(s) ;

The following statements all provide equivalent information to the analyzer:

1) KEYWORD KEY1, KEY2, KEY3;
2) THE KEYWORDS ARE: KEY1, KEY2, AND KEY3;

- KEYWORD is a required Reserved Word.
- THE, ARE and AND are Optional Reserved Words.
- KEY1, KEY2, KEY3 are names.
- The commas and semi-colon are required punctuation.
- The colon is optional punctuation.

2.2 UPL Character Set

All reserved words, names and numbers must be composed of characters in the UPL character set. The ASC II characters are classified using a code of 1 to 4, which has the following meanings:

Code 1:	Nonprinting operating system and transmission control characters to be treated as punctuation, but will always be illegal if used.
Code 2:	Punctuation, delimiters, etc. which are not allowed in names.
Code 3:	Characters allowed at any position in a name.
Code 4:	Characters allowed at any position in a name after the first.

The complete ASCII characters classified by the code are as follows:

```
CODE 1: All others

CODE 2: "&'*,:;=?|{}¬[ ]

CODE 3: ABCDEFGHIJKLMNOPQRSTUVWXYZ
        abcdefghijklmnopqrstuvwxyz
        !#$%@()

CODE 4: 0123456789
        +-./<>_
```

2.3 Words

A word in UPL is any combination of not more than thirty of the Code 3 and Code 4 characters, except that code 4 characters can not be the first character of the word.

2.4 Integer

An integer in UPL is composed of a series of digits without decimal point, plus or minus sign.

2.5 Names

All names in UPL have a type associated with them (see Appendix E for possible types). In the format for the statements, only certain types of names are allowed in certain contexts. This is indicated in the associated usage rules.

Note: Names must begin with a letter.

Note: A name in URL is any combination of not more than thirty of the above characters.
Note: Blanks may not be used in names.

2.6 Punctuation

The following characters are used for punctuation in URL:

	space (blank)
,	comma
;	semi colon
:	colon

The following rules apply to the use of punctuation in URL:
-When any punctuation appears in the format for a statement, the punctuation must be given exactly as shown.
-Two or more blanks are treated the same as a single blank.
-Blanks may be used anywhere except in words or integers.
-A colon may be used anywhere that a blank is allowed.
-A semi-colon may only be used to end a statement.

2.7 Names

Name(s) is a series of names separated by commas.

2.8 Statement Formation

Statements are formed from words and punctuation according to the rules given in chapters 3 and 4.

General rules:

- All statements must end in a semi-colon.
- Words must be separated by at least one character (punctuation, blank etc.).
- Any punctuation in the format descriptions of chapter 3 or 4 must be given exactly as shown.
- All statements, except section header statements, may be preceeded by optional name(s). The names must be used in the header statement for the section in which the statement occurs. If the name(s) are not given then the statement applies to all the names in the header statement. Alternately, if the name(s) are given, the statement will apply only to names in the list.

2.9 Sections

A problem statement in URL consists of at least one section. The possible section types are given in Appendix F. A section is a series of statements the first of which is a header statement; the type of header statement determines the type of section. The other statements in a section may be given in any order.
General rules:
-Only certain types of statements are allowed in a section, depending on the section type. The specific statements allowed in any section are given in chapter 3.

2.10 Comment-entry

Several statements have a comment-entry associated with them. Comment-entries are handled by the analyzer as follows:

-The rest of the input line containing the semi-colon after the reserved word for the statement is discarded
-Lines are read and added to the data base as given, up to and including the first line which contains a semi-colon.
-The semi-colon is replaced with a blank in this line before the line is added to the data base. (Note: then complete line is added to the data base even if the semi-colon is the first character in that line.)
-Parsing of statements begins at the first character of the following line.

2.11 Comments

For increased comprehension and documentation, comments (to be differentiated from comment-entries) can be used. Every comment must begin with /* and end with characters reversed, i.e., */. No blanks or other characters may appear between these two characters, they must be immediately adjacent. Comments are treated exactly as a blank and do not otherwise affect the analysis of the User Requirements. Although they appear in the URA As-Is-Source Listing, they are discarded by the analyzer and are not entered into the data base. The use of the dollar sign ($) in comments should be avoided, as it could have some affect on internal tracing routines within the User Requirements Analyzer.

2.12 Notation Used in Describing Syntax

In this manual, the following notation is used when describing URL 3.3 syntax.

Lower Case Words

Words written in lower case call for names to be made up and inserted by the User. The lower case descriptions of user defined names tell what kind of words the User is to make up.

Braces

When words of phrases are enclosed in braces ({ }), a choice among the two or more entries must be made. It is important to note that one of the options must be chosen. Several braces vertically on a page is equivalent to one large brace.

Brackets

Whenever notation in a model appears within brackets ([ ]), it indicates some feature the User may optionally use. Several brackets vertically on a page is equivalent to one large bracket.

Ellipsis

The ellipsis (...) signifies that the URL construct immediately preceding the ellipsis can be repeated as many times as desired by the User.

Underscoring

All upper case words which are underscored are URL Reserved Words and, if used, must appear exactly as shown. Note that when using the H6180/Multics version of URL/URA, all reserved and optional words must be in lower case.

System-Parameter

The use of system-parameter in the statement syntax denotes that the system-parameter name or integer can be used.

3. SECTION SUMMARIES

3.1 Statements Allowed in Most Sections

The following statements are allowed in almost every section:

```
     ASSERT name attribute-name attribute-value
                        [, name attribute-name attribute-value] ...;

                         { attv-name } [              { attv-name } ]
     ATTRIBUTES ARE attr-name {           } [ ,attr-name {           } ] ...
                         {  integer  } [              {  integer  } ]

     DESCRIPTION ;
             comment-entry ;

     KEYWORDS ARE keyword-name(s) ;

     RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

     SECURITY IS security-name(s) ;

     SEE-MEMO memo-name(s) ;

     SOURCE IS source-name(s) ;

     SYNONYMS ARE synonym-name(s) ;

     TRACE-KEY trace-key-name(s) ;
```

With the following exceptions:

- The RESPONSIBLE-PROBLEM-DEFINER statement is not allowed in a PROBLEM-DEFINER section.

-The SEE-KEYS statement is not allowed in the MEMO section.

-The KEYWORDS statement is not allowed in a DEFINE section for a KEYWORD.

-The SOURCE statement is not allowed in a DEFINE section for a SOURCE.

-The SECURITY statement is not allowed in a DEFINE section for a SECURITY.

-The TRACE-KEY statement is not allowed in a DEFINE section for a TRACE-KEY.

-No statements are allowed in a DESIGNATE section.

SECTION SUMMARIES

3.2 CONDITION Section

```
CONDITION name(s) ;


ASSERT name attribute-name attribute-value

             [, name attribute-name attribute-value] ...;


                       { attv-name } [                  { attv-name } ]
ATTRIBUTES ARE attr-name {         } [ ,attr-name {             } ]...
                       {  integer  } [                  {  integer  } ]


          { TRUE  }
BECOMING  {       } CAUSES event-name(s);
          { FALSE }


          { TRUE  }
BECOMING  {       } INTERRUPTS process-name(s);
          { FALSE }


          { TRUE  }
BECOMING  {       } TERMINATES process-name(s);
          { FALSE }


          { TRUE  }
BECOMING  {       } TRIGGERS process-name(s);
          { FALSE }

                  input-
                  output-
DEPENDS ON        element-    name(s)  ;
                  entity-
                  group-
                  set-


DESCRIPTION ;
      comment-entry ;


KEYWORDS ARE keyword-name(s) ;


     { TRUE  }           event-
MADE {       } BY        input-   name(s)
```

```
    { FALSE }         process-

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;

{ TRUE  }
{       } WHILE ;
{ FALSE }
     comment-entry ;
```

3.3 DEFINE Section

```
                {ATTRIBUTE             }[          {ATTRIBUTE            } ]
                {ATTRIBUTE-VALUE       }[          {ATTRIBUTE-VALUE      } ]
                {CLASSIFICATION        }[          {CLASSIFICATION       } ]
                {KEYWORD               }[          {KEYWORD              } ]
                {MAILBOX               }[          {MAILBOX              } ]
DEFINE name     {SECURITY              }[, name    {SECURITY             } ] ... ;
                {SOURCE                }[          {SOURCE               } ]
                {SUBSETTING-CRITERION  }[          {SUBSETTING-CRITERION } ]
                {SYSTEM-PARAMETER      }[          {SYSTEM-PARAMETER     } ]
                {TRACE-KEY             }[          {TRACE-KEY            } ]
```

```
APPLIES TO name(s) ;


ASSERT name attribute-name attribute-value

               [, name attribute-name attribute-value] ...;


                               { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name       {           } [ ,attr-name      {           } ] ...
                               {  integer  } [                 {  integer  } ]


DESCRIPTION ;
     comment-entry ;


KEYWORDS ARE keyword-name(s) ;


MAINTAINED BY process-name(s)

     [DEPENDING ON element-     name(s) ]
     [             condition-           ]

     [             group-               ]
     [             entity-              ]
     [FOR EACH     element-     name(s) ];
     [             output-              ]
     [             input-               ]
     [             set-                 ]


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;
```

```
SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SUBSETTING-CRITERION FOR set-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;


             {               integer                   }
             {                                         }
VALUES ARE   { { *min    }              {   *max   }  } ;
             { {         }     THRU     {          }  }
             { { NEGINF  }              {  POSINF  }  }
```

* Min and max must be non-negative integers.

3.4 DESIGNATE Section

```
DESIGNATE name AS A SYNONYM FOR name
     [ , name AS A SYNONYM FOR name ] ... ;
```

3.5 ELEMENT Section

```
ELEMENT names(s) ;


ASSERT name attribute-name attribute-value

                [, name attribute-name attribute-value] ...;

ASSOCIATED WITH relation-name(s) ;


                           [ attv-name } [                 [ attv-name } ]
ATTRIBUTES ARE attr-name   [             } [  ,attr-name [             } ] ..
                           [   integer   } [                 [   integer   } ]


CLASSIFICATION classification-name [ integer ]

                [, classification-name [ integer ]]... ;

                group-
               entity-
CONTAINED IN    input-name(s)  ;
               output-


                                      [           group-               ]
                                      [          entity-               ]
DERIVED BY  process-name(s)  [  USING          set-      name(s) ]
                                      [           input-               ]
                                      [         element-               ]

      [DEPENDING ON element-      name(s) ]
      [             condition-            ]

      [             group-                ]
      [             entity-               ]
      [FOR EACH     element-      name(s) ];
      [             output-               ]
      [             input-                ]
      [             set-                  ]


DESCRIPTION ;
      comment-entry ;
```

```
IDENTIFIES entity-name(s) ;

KEYWORDS ARE keyword-name(s) ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SUBSETTING-CRITERION FOR set-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;

                             [         group-             ]
                             [        entity-             ]
UPDATED BY  process-name(s)  [ USING  element-    name(s) ]
                             [         input-             ]
                             [           set-             ]

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]
```

```
                                [                       set-              ]
                                [         { DERIVE }  *output-            ]
USED BY process-name(s)   [  TO  {            }     entity-    name(s) ]
                                [         { UPDATE }    group-            ]
                                [                     element-            ]

      [DEPENDING ON element-      name(s) ]
      [             condition-            ]

      [             group-                ]
      [             entity-               ]
      [FOR EACH     element-      name(s) ];
      [             output-               ]
      [             input-                ]
      [             set-                  ]
```

* Output-name(s) may only be used with the DERIVE clause.

```
             {             integer                      }
             {                                          }
VALUES ARE   { { *min    }          {  *max    } }  ;
             { {         }   THRU   {          } }
             { { NEGINF  }          { POSINF   } }
```

* Min and max must be integers.

3.6 ENTITY Section

```
     ENTITY name(s) ;


     ASSERT name attribute-name attribute-value

                   [, name attribute-name attribute-value] ...;


                             { attv-name } [                { attv-name }  ]
     ATTRIBUTES ARE attr-name {           } [  ,attr-name {               }  ].
                             {  integer  } [                {  integer  }  ]

     CARDINALITY IS system-parameter ;


     CLASSIFICATION classification-name [ integer ]

                   [, classification-name [ integer ]]... ;
                                            element-
     CONSISTS OF [ system-parameter ]  group-name
                                                element-
                   [ , [ system-parameter ]  group-name ] ... ;


     CONTAINED IN  set-name(s) ;

                                    [             group-              ]
                                    [            entity-              ]
     DERIVED BY  process-name(s) [ USING        set-    name(s) ]
                                    [             input-              ]
                                    [           element-              ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

```
DESCRIPTION ;
    comment-entry ;


                  group
IDENTIFIED BY element-name(s) ;


KEYWORDS ARE keyword-name(s) ;


RELATED TO entity-name VIA relation-name ;


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;


SECURITY IS security-name(s) ;


SEE-MEMO memo-name(s) ;


SOURCE IS source-name(s) ;


SYNONYMS ARE synonym-name(s) ;


TRACE-KEY trace-key-name(s) ;


                                  [         group-         ]
                                  [        entity-         ]
UPDATED BY  process-name(s) [ USING element-     name(s) ]
                                  [          input-         ]
                                  [            set-         ]

     [DEPENDING ON element-     name(s) ]
     [             condition-           ]

     [             group-               ]
     [             entity-              ]
     [FOR EACH     element-     name(s) ];
     [             output-              ]
     [             input-               ]
     [             set-                 ]
```

```
                               [                       set-              ]
                               [        { DERIVE }  *output-             ]
USED BY process-name(s)  [ TO  {          }    entity-     name(s) ]
                               [        { UPDATE }     group-            ]
                               [                      element-           ]

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR EACH     element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]

* Output-name(s) may only be used with the DERIVE clause.


VOLATILITY ;
     comment-entry ;
```

2.7 EVENT Section

```
EVENT name(s) ;


ASSERT name attribute-name attribute-value

              [, name attribute-name attribute-value] ...;


                              { attv-name } [              { attv-name }  ]
ATTRIBUTES ARE attr-name      {           } [  ,attr-name  {             } ] ..
                              {  integer  } [              {   integer   } ]


              event-
CAUSED BY               name(s)
              input-

     [DEPENDING ON element-     name(s) ]
     [             condition-           ]

     [             group-               ]
     [             entity-              ]
     [FOR EACH     element-     name(s) ];
     [             output-              ]
     [             input-               ]
     [             set-                 ]


                                           { TRUE  }
CAUSED WHEN condition-name BECOMES         {       };
                                           { FALSE }


CAUSES event-name(s)

     [DEPENDING ON element-     name(s) ]
     [             condition-           ]

     [             group-               ]
     [             entity-              ]
     [FOR EACH     element-     name(s) ];
     [             output-              ]
     [             input-               ]
     [             set-                 ]
```

```
DESCRIPTION :
      comment-entry ;

        {system-parameter TIMES-PER interval-name}
HAPPENS {EVERY system-parameter interval-name      };
        {[WITHIN] system-parameter interval-name   }
        {              AFTER event-name            }

ON INCEPTION OF process-name(s) ;

INTERRUPTS process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]

KEYWORDS ARE keyword-name(s) ;

                            { TRUE  }
MAKES condition-name(s)     {       }
                            { FALSE }

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;
```

SECTION SUMMARIES

```
SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TERMINATES process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]

ON TERMINATION OF process-name(s) ;

TRACE-KEY trace-key-name(s) ;

TRIGGERS process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]
```

3.8 GROUP Section

```
GROUP name(s) ;


ASSERT name attribute-name attribute-value

              [, name attribute-name attribute-value] ...;


ASSOCIATED WITH relation-name(s) ;


                              { attv-name } [               { attv-name }  ]
ATTRIBUTES ARE attr-name      {             } [ ,attr-name  {             }  ] ..
                              {  integer   } [               {  integer   }  ]


CLASSIFICATION classification-name [ integer ]

              [, classification-name [ integer ]]... ;

                                     element-
CONSISTS OF [ system-parameter ]   group-name

                                          element-
              [ , [ system-parameter ]   group-name ] ... ;



              group-
              entity-
CONTAINED IN  input-   name(s) ;
              output-
```

SECTION SUMMARIES

```
                                     [          group-              ]
                                     [         entity-              ]
DERIVED BY  process-name(s)  [ USING         set-     name(s) ]
                                     [          input-              ]
                                     [        element-              ]

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

DESCRIPTION ;
     comment-entry ;

IDENTIFIES entity-name(s)  ;

KEYWORDS ARE keyword-name(s)  ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s)  ;

SEE-MEMO memo-name(s)  ;

SOURCE IS source-name(s)  ;

SUBSETTING-CRITERION FOR set-name(s)  ;

SYNONYMS ARE synonym-name(s)  ;

TRACE-KEY trace-key-name(s)  ;
```

```
                                 [         group-             ]
                                 [        entity-             ]
UPDATED BY  process-name(s)  [  USING  element-    name(s) ]
                                 [          input-            ]
                                 [            set-            ]

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR  EACH    element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]


                                 [                        set-              ]
                                 [       { DERIVE }  *output-               ]
USED BY process-name(s) [  TO {        }    entity-    name(s) ]
                                 [       { UPDATE }     group-              ]
                                 [                   element-               ]

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR  EACH    element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]
```

* Output-name(s) may only be used with the DERIVE clause.

3.9 INPUT Section

```
INPUT name(s) ;


ASSERT name attribute-name attribute-value

              [, name attribute-name attribute-value] ...;


                              { attv-name } [                   { attv-name }  ]
ATTRIBUTES ARE attr-name      {           } [  ,attr-name       {           }  ] ...
                              {  integer  } [                   {  integer  }  ]


CAUSES event-name(s)

      [DEPENDING ON  element-    name(s) ]
      [              condition-          ]

      [              group-              ]
      [              entity-             ]
      [FOR EACH      element-    name(s) ];
      [              output-             ]
      [              input-              ]
      [              set-                ]


CLASSIFICATION classification-name [ integer ]

              [, classification-name [ integer ]]... ;


                                          element-
CONSISTS OF [ system-parameter ]   group-name

                                                  element-
              [ , [ system-parameter ]   group-name ] ... ;


CONTAINED IN set-name(s) ;


DESCRIPTION ;
      comment-entry ;
```

```
GENERATED BY interface-name(s)

      [DEPENDING ON element-    name(s) ]
      [             condition-          ]

      [             group-              ]
      [             entity-             ]
      [FOR EACH     element-    name(s) ];
      [             output-             ]
      [             input-              ]
      [             set-                ]

        {system-parameter TIMES-PER interval-name}
HAPPENS {EVERY system-parameter interval-name    };
        {[WITHIN] system-parameter interval-name }
        {               AFTER event-name         }


INTERRUPTS process-name(s)

      [DEPENDING ON element-    name(s) ]
      [             condition-          ]

      [             group-              ]
      [             entity-             ]
      [FOR EACH     element-    name(s) ];
      [             output-             ]
      [             input-              ]
      [             set-                ]

KEYWORDS ARE keyword-name(s) ;
```

```
                     { TRUE  }
MAKES condition-name(s) {       }
                     { FALSE }

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

PART OF input-name ;


RECEIVED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SUBPARTS ARE input-name(s) ;

SYNONYMS ARE synonym-name(s) ;
```

```
TERMINATES process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

TRACE-KEY trace-key-name(s) ;

TRIGGERS process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

                               [                       set-             ]
                               [      { DERIVE }  *output-              ]
USED BY process-name(s) [  TO  {          }   entity-     name(s) ]
                               [      { UPDATE }    group-              ]
                               [                  element-              ]

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

* Output-name(s) may only be used with the DERIVE clause.

3.10 INTERFACE Section

```
INTERFACE name(s) ;


ASSERT name attribute-name attribute-value

             [, name attribute-name attribute-value] ...;


                            { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name    {           } [  ,attr-name     {           } ]..
                            {  integer  } [                 {  integer  } ]


DESCRIPTION :
     comment-entry :


GENERATES input-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


KEYWORDS ARE keyword-name(s) ;


PART OF interface-name ;


RECEIVES output-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

```
RESPONSIBLE FOR set-name(s) ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SECURITY-ACCESS-RIGHT classification-name [ integer ]
             [, classification-name [ integer ]]... ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SUBPARTS ARE interface-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.11 INTERVAL Section

```
INTERVAL name(s) ;

ASSERT name attribute-name attribute-value
            [, name attribute-name attribute-value] ...;

                      { attv-name } [             { attv-name } ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name {           } ] ..
                      {  integer  } [             {  integer  } ]

CONSISTS OF [ system-parameter ] interval-name
            [ , [ system-parameter ] interval-name ] ... ;

DESCRIPTION :
      comment-entry ;

KEYWORDS ARE keyword-name(s) ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.12 MEMO Section

```
MEMO name(s) ;

APPLIES TO non-memo-name(s) ;

ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...;

                             { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name     {           } [  ,attr-name     {           } ] ;
                             {  integer  } [                 {  integer  } ]

DESCRIPTION ;
     comment-entry ;

KEYWORDS ARE keyword-name(s) ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.13 OUTPUT Section

```
OUTPUT name(s) ;


ASSERT name attribute-name attribute-value

                [, name attribute-name attribute-value] ...;


                      { attv-name } [                 { attv-name }  ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name {             }  ] ...
                      {   integer  } [                {  integer   }  ]


CLASSIFICATION classification-name [ integer ]

                [, classification-name [ integer ]]... ;

                                       element-
CONSISTS OF [ system-parameter ]   group-name

                                            element-
                [ , [ system-parameter ]   group-name ] ... ;


CONTAINED IN set-name(s) ;


                                 [              group-          ]
                                 [             entity-          ]
DERIVED BY   process-name(s) [ USING        set-     name(s) ]
                                 [              input-          ]
                                 [            element-          ]

        [ DEPENDING ON element-     name(s) ]
        [              condition-           ]

        [              group-               ]
        [              entity-              ]
        [ FOR EACH     element-     name(s) ];
        [              output-              ]
        [              input-               ]
        [              set-                 ]


DESCRIPTION :
      comment-entry ;
```

```
GENERATED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

        {system-parameter TIMES-PER interval-name}
HAPPENS {EVERY system-parameter interval-name    };
        {[WITHIN] system-parameter interval-name }
        {                AFTER event-name        }


KEYWORDS ARE keyword-name(s) ;


PART OF output-name ;


RECEIVED BY interface-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;


SECURITY IS security-name(s) ;


SEE-MEMO memo-name(s) ;


SOURCE IS source-name(s) ;


SUBPARTS ARE output-name(s) ;
```

<u>SYNONYMS</u> ARE synonym-name(s) ;

<u>TRACE-KEY</u> trace-key-name(s) ;

3.14 PROBLEM-DEFINER Section

```
PROBLEM-DEFINER name(s) ;


ASSERT name attribute-name attribute-value
                  [, name attribute-name attribute-value] ...;

                           { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name {             } [  ,attr-name {             } ] .
                           {  integer  } [                 {  integer  } ]

DESCRIPTION :
     comment-entry ;

KEYWORDS ARE keyword-name(s) ;

MAILBOX IS mailbox-name ;

RESPONSIBLE FOR name(s) ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.15 PROCESS Section

```
PROCESS name(s) ;


ASSERT name attribute-name attribute-value

                [, name attribute-name attribute-value] ...;

                           { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name   {           } [  ,attr-name     {             } ] ...
                           {  integer  } [                 {  integer    } ]


               set-          [            set-            ]
            output-          [          input-            ]
DERIVES   element-name(s)    [ USING element-name(s)      ]
            entity-          [          entity-           ]
             group-          [           group-           ]

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR EACH     element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]


DESCRIPTION ;
     comment-entry ;


GENERATES output-name(s)

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR EACH     element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]
```

```
        {system-parameter TIMES-PER interval-name}
HAPPENS {EVERY system-parameter interval-name    };
        {[WITHIN] system-parameter interval-name }
        {                AFTER event-name        }


INCEPTION-CAUSES event-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]


                   event-
INTERRUPTED BY     input-name(s)
                  process-

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]


                                              { TRUE  }
    INTERRUPTED WHEN condition-name BECOMES   {       };
                                              { FALSE }


    INTERRUPTS process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]
```

```
KEYWORDS ARE keyword-name(s) ;

                        relation-
MAINTAINS subsetting-criteria-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


                              { TRUE  }
MAKES condition-name(s)       {       }
                              { FALSE }

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


PART OF process-name ;

PERFORMED BY processor-name ;

PROCEDURE ;
     comment-entry ;
```

```
RECEIVES input-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

RESOURCE-USAGE :

          system-parameter FOR resource-usage-parameter-name;


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SECURITY-ACCESS-RIGHT classification-name [ integer ]

               [, classification-name [ integer ]]... ;

SEE-MEMO memo-name(s) ;


SOURCE IS source-name(s) ;

SUBPARTS ARE process-name(s) ;

SYNONYMS ARE synonym-name(s) ;
```

```
                  event-
TERMINATED BY     input-name(s)
                  process-

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


                                          { TRUE  }
TERMINATED WHEN condition-name BECOMES    {       }  ;
                                          { FALSE }


TERMINATES process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

TERMINATION-CAUSES event-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

TRACE-KEY trace-key-name(s)  ;
```

```
                    event-
TRIGGERED BY        input-name(s)
                  process-

        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]

                                                { TRUE  }
TRIGGERED WHEN condition-name BECOMES           {       }
                                                { FALSE }

        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]

TRIGGERS process-name(s)


        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]
```

```
                   group-                [        group-             ]
                  entity-                [       entity-             ]
        UPDATES  element-name(s)         [ USING element-    name(s) ]
                     set-                [          set-             ]
                                         [        input-             ]

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]

             set-            [                    set-             ]
           input-            [    { DERIVE }  *output-             ]
   USES  element-name(s)     [ TO {        }   element-    name(s) ]
           group-            [    { UPDATE }    group-             ]
          entity-            [                 entity-             ]

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]

 * Output-name(s) may only be used with the DERIVE clause.


 UTILIZED BY process-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

```
UTILIZES process-name(s)

     [DEPENDING ON element-     name(s) ]
     [             condition-           ]

     [             group-               ]
     [             entity-              ]
     [FOR EACH     element-     name(s) ];
     [             output-              ]
     [             input-               ]
     [             set-                 ]
```

3.16 PROCESSOR Section

```
PROCESSOR processor-name(s);


ASSERT name attribute-name attribute-value

                 [, name attribute-name attribute-value] ...;


                        { attv-name } [               { attv-name } ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name  {           } ] ..
                        {  integer  } [               {  integer  } ]


CONSUMES resource-name AT RATE OF

           system-parameter PER resource-usage-parameter-name;


DESCRIPTION ;
     comment-entry ;

KEYWORDS ARE keyword-name(s) ;

PART OF processor-name;

PERFORMS process-name(s);

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SECURITY-ACCESS-RIGHT classification-name [ integer ]

               [, classification-name [ integer ]]... ;

SEE-MEMO memo-name(s) ;
```

<u>SOURCE</u> IS source-name(s) ;

<u>SUBPARTS</u> ARE processor-name(s);

<u>SYNONYMS</u> ARE synonym-name(s) ;

<u>TRACE-KEY</u> trace-key-name(s) ;

3.17 RELATION Section

```
RELATION name(S) ;


ASSERT name attribute-name attribute-value
                 [, name attribute-name attribute-value] ...;


                     group-
ASSOCIATED-DATA IS element-name(S) ;


                            { attv-name } [                 { attv-name }  ]
ATTRIBUTES ARE attr-name    {             } [  ,attr-name {               }  ] ..
                            {   integer   } [                 {   integer   }  ]


BETWEEN entity-name AND entity-name ;


CARDINALITY IS system-parameter ;


CONNECTIVITY IS system-parameter TO system-parameter ;


DERIVATION ;
      comment-entry ;


DESCRIPTION ;
      comment-entry ;


KEYWORDS ARE keyword-name(S) ;
```

```
MAINTAINED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.18 RESOURCE Section

```
RESOURCE resource-name(s) ;


ASSERT name attribute-name attribute-value

                 [, name attribute-name attribute-value] ...;


                                  { attv-name } [               { attv-name }  ]
ATTRIBUTES ARE attr-name          {           } [ ,attr-name    {             }  ] ..
                                  {  integer  } [               {  integer    }  ]



CONSUMED BY processor-name(s) AT RATE OF

           system-parameter PER resource-usage-parameter-name;



DESCRIPTION ;
       comment-entry ;


KEYWORDS ARE keyword-name(s) ;


MEASURED IN unit-name;


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;


SECURITY IS security-name(s) ;


SEE-MEMO memo-name(s) ;


SOURCE IS source-name(s) ;


SYNONYMS ARE synonym-name(s) ;


TRACE-KEY trace-key-name(s) ;
```

3.19 RESOURCE-USAGE-PARAMETER Section

```
RESOURCE-USAGE-PARAMETER resource-usage-parameter-name(s);


ASSERT name attribute-name attribute-value

                  [, name attribute-name attribute-value] ...;

                             { attv-name } [                  { attv-name } ]
ATTRIBUTES ARE attr-name     {           } [ ,attr-name       {           } ] .
                             {  integer  } [                  {  integer  } ]

DESCRIPTION ;
      comment-entry ;

KEYWORDS ARE keyword-name(s) ;

RESOURCE-USAGE-PARAMETER-VALUE :

                  system-parameter FOR process-name;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;

SYNONYMS ARE synonym-name(s) ;

TRACE-KEY trace-key-name(s) ;
```

3.20 SET Section

```
SET name(S) :


ASSERT name attribute-name attribute-value

             [, name attribute-name attribute-value] ...;

                            { attv-name } [              { attv-name } ]
ATTRIBUTES ARE attr-name    {           } [ ,attr-name   {           } ]..
                            {  integer  } [              {  integer  } ]


CARDINALITY IS system-parameter ;


CLASSIFICATION classification-name [ integer ]

             [, classification-name [ integer ]]... :


                                          input-
CONSISTS OF [ system-parameter ]          output-name
                                          entity-

                                                 input-
             [ , [ system-parameter ]           output-name ] ... ;
                                                 entity-


DERIVATION :
        comment-entry ;
```

```
                            [          group-          ]
                            [         entity-          ]
DERIVED BY  process-name(s) [ USING      set-  name(s) ]
                            [          input-          ]
                            [        element-          ]

     [ DEPENDING ON element-    name(s) ]
     [              condition-          ]

     [              group-              ]
     [              entity-             ]
     [ FOR EACH     element-    name(s) ];
     [              output-             ]
     [              input-              ]
     [              set-                ]

DESCRIPTION ;
      comment-entry ;

KEYWORDS ARE keyword-name(s) ;

RESPONSIBLE-INTERFACE IS interface-name(s) ;

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

SECURITY IS security-name(s) ;

SEE-MEMO memo-name(s) ;

SOURCE IS source-name(s) ;


SUBSET OF set-name(s) ;

SUBSETS ARE set-name(s) ;

                                          group-
SUBSETTING-CRITERIA ARE                 element-   name(s) ;
                        subsetting-criterion-

SYNONYMS ARE synonym-name(s) ;
```

```
TRACE-KEY trace-key-name(s) ;

                               [         group-             ]
                               [         entity-            ]
UPDATED BY  process-name(s)  [ USING  element-   name(s) ]
                               [         input-             ]
                               [         set-               ]

     [DEPENDING ON element-    name(s) ]
     [              condition-         ]

     [              group-             ]
     [              entity-            ]
     [FOR EACH      element-   name(s) ];
     [              output-            ]
     [              input-             ]
     [              set-               ]

                                  [                       set-             ]
                                  [    { DERIVE } *output-             ]
USED BY process-name(s)  [  TO {        }  entity-    name(s) ]
                                  [    { UPDATE }  group-              ]
                                  [                  element-           ]

     [DEPENDING ON element-    name(s) ]
     [              condition-         ]

     [              group-             ]
     [              entity-            ]
     [FOR EACH      element-   name(s) ];
     [              output-            ]
     [              input-             ]
     [              set-               ]

* Output-name(s) may only be used with the DERIVE clause.

VOLATILITY-MEMBER :
        comment-entry ;

VOLATILITY-SET :
        comment-entry ;
```

3.21 UNIT Section

```
UNIT name(s);


ASSERT name attribute-name attribute-value

                  [, name attribute-name attribute-value] ...;

                         { attv-name } [                  { attv-name } ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name {             } ] ...
                         {  integer  } [                  {  integer  } ]

DESCRIPTION ;
      comment-entry ;


KEYWORDS ARE keyword-name(s) ;


MEASURES resource-name(s) ;


RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;


SECURITY IS security-name(s) ;


SEE-MEMO memo-name(s) ;


SOURCE IS source-name(s) ;


SYNONYMS ARE synonym-name(s) ;


TRACE-KEY trace-key-name(s) ;
```

4. INDIVIDUAL STATEMENTS

The following pages give a description of all allowable URL statements. With each statement there is a declaration of purpose, the syntax, complementary statements (if any exist), and the rules concerning the type of names allowed in the syntax and restrictions pertaining to the statement. Each page is intended to be a unit by itself; all the information needed for a statement is given on the page for that statement. Therefore, the same information may be given on several different pages.

The statements are listed alphabetically. Statements that may occur in several sections are arranged alphabetically by section type.

4.1 CONDITION Section Header Statement

Purpose:

To indicate a TRUE/FALSE state within the system, and to optionally link that state to EVENTS and/or the initiation of PROCESSES. Thus the analyst has a way to indicate a processing path to be followed when one or more CONDITIONS are satisfied, or alternative processing paths when CONDITIONS are not met.

Syntax:

__CONDITION__ condition-name(s) ;

Usage Rules:

-Must be the first statement in a CONDITION section.

-More than one CONDITION can be defined at a time.

Synonyms:

COND CONDITIONS

Examples:

- CONDITION PAYCHECK-DISTRIBUTED;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
     ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...:
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
     - ASSERT data-name-1 type character;

     - ASRT sine-function arguments 1,
            coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                            { attv-name } [             { attv-name } ]
ATTRIBUTES ARE attr-name {             } [ ,attr-name {             } ]...
                            {  integer  } [             {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR	ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR Z7Z9V9;

BECOMING CAUSES Statement

Purpose:
To specify the EVENT(S) caused by this CONDITION.

Syntax:

```
          { TRUE  }
BECOMING  {       } CAUSES event-name(s);
          { FALSE }
```

Complementary Statements:
CAUSED statement in the EVENT section.

Usage Rules:
- A CONDITION BECOMING TRUE or FALSE may CAUSE several different EVENTS.

- A CONDITION BECOMING TRUE may CAUSE one set of EVENTS and BECOMING FALSE may CAUSE a second set.

Synonyms:

```
{ BEC  }
{      } CSS
{ BECG }
```

Examples:

- BECOMING FALSE CAUSES ERROR-DETECTED ;

- BECOMING TRUE CAUSES SUBPROCESS-COMPLETION, MAIN-PROCESS-COMPLETION ;

- BEC T CSS EVENT-1, EVENT-2, EVENT-3 ;

- BECG F CSS TIME-CARD-RECOGNIZED ;

BECOMING INTERRUPTS Statement

Purpose:

To specify the PROCESS(ES) interrupted by a change of state for this CONDITION.

Syntax:

```
                { TRUE  }
     BECOMING   {       } INTERRUPTS process-name(s) ;
                { FALSE }
```

Complementary Statements:

INTERRUPTED statement in the PROCESS section.

Usage Rules:

- A CONDITION BECOMING TRUE or FALSE may INTERRUPT several PROCESSES.

- A CONDITION BECOMING TRUE may INTERRUPT one set of PROCESSES and BECOMING FALSE may INTERRUPT a second set.

Synonyms:

```
     { BEC  }
     {      } INTS
     { BECG }
```

Examples:

```
     - BECOMING FALSE INTERRUPTS NORMAL-PROCESSING ;

     - BEC T INTS PACK-FOR-SHIPPING, BILLING ;

     - BECG F INTS SALARY-COMPUTATION ;
```

BECOMING TERMINATES Statement

Purpose:

To specify a PROCESS/PROCESSES that are terminated when this CONDITION enters a given state.

Syntax:

```
         { TRUE  }
BECOMING {       } TERMINATES process-name(s) ;
         { FALSE }
```

Complementary Statements:

TERMINATED statement in PROCESS section.

Usage Rules:

- A CONDITION BECOMING TRUE or FALSE may TERMINATE several PROCESSES.

- A CONDITION BECOMING TRUE may TERMINATE one set of PROCESSES and BECOMING FALSE may TERMINATE a second set.

Synonyms:

```
{ BEC  }
{      } TRMS
{ BECG }
```

Examples:

- BECOMING TRUE TERMINATES BILLING-PROCESS ;

- BEC T TERMINATES SALARIED-PAY-COMPUTATION, HOURLY-PAY-COMPUTATION ;

- BECG F TRMS ERROR-HANDLER ;

BECOMING TRIGGERS Statement

Purpose:

To specify a PROCESS/PROCESSES that are triggered by a change in state for this CONDITION.

Syntax:

```
                { TRUE  }
      BECOMING  {       } TRIGGERS process-name(s);
                { FALSE }
```

Complementary Statements:

TRIGGERED statement in the PROCESS section.

Usage Rules:

- A CONDITION BECOMING TRUE or FALSE may TRIGGER several PROCESSES.

- A CONDITION BECOMING TRUE may TRIGGER one set of PROCESSES and BECOMING FALSE may TRIGGER a second set.

Synonyms:

```
      { BEC  }
      {      } TRGS
      { BECG }
```

Examples:

- BECOMING TRUE TRIGGERS BILLING-PROCESS ;

- BEC T TRIGGERS SALARIED-PAY-COMPUTATION, HOURLY-PAY-COMPUTATION ;

- BECG F TRGS ERROR-HANDLER ;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
          comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
    THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
    ANY RELEVANT INFORMATION GOES HERE;
```

DEPENDS ON Statement

Purpose:
To declare interdependencies among conditions and data.

Syntax:

```
                      input-
                      output-
      DEPENDS ON      element-    name(s) ;
                      entity-
                      group-
                      set-
```

Complementary Statements:
None.

Usage Rules:
- A section may have several DEPENDS statements.

Synonyms:
DPND DPNS

Examples:

```
DPNS INPUT-A;
DPND ON ELE-A; ELE-B;
```

KEYWORD Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

KEYWORDS ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a KEYWORD.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MADE Statement

Purpose:

To specify those EVENT(S), INPUT(S), and PROCESS(ES) which may set this CONDITION, to indicate the value to which it is set, and to express any conditions and/or iterations associated with the action.

Syntax:

```
         { TRUE  }          event-
MADE     {       } BY       input-    name(s)
         { FALSE }        process-

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:

MAKES statement in EVENT, INPUT, and PROCESS sections.

Usage Rules:

- A CONDITION may be set by several EVENTS.

- A CONDITION may be MADE TRUE by one set of EVENTS and MADE FALSE by another set of EVENTS.

Synonyms:

DPNG DPG FC

Examples:

- MADE FALSE BY INPUT-ARRIVAL DEPENDING ON NO-TIME-CARD-FOUND;

- MADE FALSE BY INPUT-ERROR, PROCESSING-ERROR

```
          FOR EC TIME-CARD;

  - MADE T ERROR-OCCURRENCE
          DPNG ON INPUT-ERROR, PROCESSING-ERROR
          FOR EC INPUT-A;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:
To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

<u>SECURITY</u> IS security-name(s) ;

Complementary Statements:
APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

-A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO RW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms (e.g. Abbreviations) for section names in the documentation. A synonym can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section .

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE C-11, CONDITION-11;
- SYNONYM IS CONDITION-11;
- SYN ALPHA;

<u>TRACE-KEY Statement</u>

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

<u>TRACE-KEY</u> trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

WHILE Statement

Purpose:
    To give an expression on which this CONDITION depends.

Syntax:

```
      { TRUE  }
      {       } WHILE ;
      { FALSE }
            comment-entry ;
```

Complementary Statements:
    None.

Usage Rules:
    -May be given only once for any CONDITION.

Synonyms:

```
          {   T   }
          {       }  WHL
          {   F   }
```

Examples::

```
     - TRUE WHILE;
        STILL AN EMPLOYEE;

     - FALSE WHILE;
        SYSTEM-BEING-UPDATED;

     - T WHL;
        SYSTEM OUTPUT STILL VALID;
```

4.2 DEFINE Section Header Statement

Purpose:

To describe in greater detail certain name types within URL. For example, if one wished to show a value or range of values for a system parameter, it would be done in this section.

Syntax:

```
              {ATTRIBUTE              }[         {ATTRIBUTE              } ]
              {ATTRIBUTE-VALUE        }[         {ATTRIBUTE-VALUE        } ]
              {CLASSIFICATION         }[         {CLASSIFICATION         } ]
              {KEYWORD                }[         {KEYWORD                } ]
              {MAILBOX                }[         {MAILBOX                } ]
DEFINE name   {SECURITY               }[, name   {SECURITY               } ] ... ;
              {SOURCE                 }[         {SOURCE                 } ]
              {SUBSETTING-CRITERION   }[         {SUBSETTING-CRITERION   } ]
              {SYSTEM-PARAMETER       }[         {SYSTEM-PARAMETER       } ]
              {TRACE-KEY              }[         {TRACE-KEY              } ]
```

Usage Rules:

-It must be the first statement in the DEFINE section.

-Several names may be defined at once.

Synonyms:

```
        { ATTR                                   }
        { ATTV                                   }
        { CLS   CLASSIFICATIONS                  }
        { KEY                                    }
        { BOX   MBX                              }
    DEF { SEC                                    }
        { SRC                                    }
        { SSCN                                   }
        { SYSP   SYSPAR   SYSTEM-PARAMETERS      }
        { TKEY                                   }
```

Examples:

```
- DEFINE NAME-A ATTRIBUTE .............DEF NAME-A ATTR
- DEFINE NAME-B ATTRIBUTE-VALUE .......DEF NAME-B ATTV
- DEFINE NAME-C CLASSIFICATION ........DEF NAME-C CLS
- DEFINE NAME-D KEYWORD ...............DEF NAME-D KEY
- DEFINE NAME-E MAILBOX ...............DEF NAME-E MBX
- DEFINE NAME-F SECURITY ..............DEF NAME-F SEC
- DEFINE NAME-G SOURCE ................DEF NAME-G SRC
- DEFINE NAME-H SUBSETTING-CRITERION ..DEF NAME-H SSCN
- DEFINE NAME-I SYSTEM-PARAMETER ......DEF NAME-I SYSP
- DEFINE NAME-J TRACE-KEY .............DEF NAME-J TKEY
```

APPLIES Statement

Purpose:
To tie the information contained in the DEFINE section to any new or revised sections to which it applies.

Syntax:

APPLIES TO name(s) ;

Complementary Statements:
KEYWORDS, MAILBOX, SECURITY, SOURCE AND TRACE-KEY statements.

Usage Rules:
- This statement may only be given in the DEFINE sections for those names which are of the type KEYWORD, SECURITY, SOURCE, MAILBOX, or TRACE-KEY.
- The statement may be given as many times an necessary for the name.
- Multiple APPLIES statements for the same name are equivalent to a single statement with all the names in the list.

Synonyms:

APP

Examples:

```
-APPLIES TO NETWORK-IDENT;

-APPLIES TO NETWORK-IDENT,COMPANY-AND-AREA,TYPE-MATERIAL;

-APP PROCESS-1;

-APP TO NETWORK-IDENT, COMPANY-AND-AREA, TYPE-MATERIAL;
```

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
              [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                            { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name    {           } [ ,attr-name      {           } ]...
                            {  integer  } [                 {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORD Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for a KEYWORD.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

AD-A060 780 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78 F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-130-VOL-2 NL

2 OF 6
AD
A060780

MAINTAINED Statement

Purpose:

To give the PROCESSES which maintain a SUBSETTING-CRITERION, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
MAINTAINED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

MAINTAINS statement in PROCESS section.

Usage Rules:

-A SUBSETTING-CRITERION can be MAINTAINED by more than one PROCESS.

-THIS STATEMENT MAY ONLY BE USED TO DESCRIBE subsetting-criterion NAMES.

Synonyms:

DPNG DPG EC

Examples:

```
     - MAINTAINED BY FIRST-PROCESS;

     - MIND PROCESS-A, PROCESS-B
              DEPENDING ON ERROR-OCCURRENCE
              FOR EACH INPUT-1, INPUT-B;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:
RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

-A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SUBSETTING-CRITERION Statement

Purpose:

To indicate that this name is used to extract information from a SET to produce a SUBSET.

Syntax:

<u>SUBSETTING-CRITERION</u> FOR set-name(s) ;

Complementary Statements:

SUBSETTING-CRITERIA statement in a SET section.

Usage Rules:

-The names must be SET names.

-This statement may only be used to describe SUBSETTING-CRITERION names. -A name so defined may be a SUBSETTING-CRITERION for more than one SET.

Synonyms:

SSCN

Examples:

- SUBSETTING-CRITERION FOR SET-GROUP-BANKS, SET-GROUP-CKTS;
- SSCN: FILE-107, FILE-108;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms (e.g. Abbreviations) for section names in the documentation. A synonym can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section .

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE ATTR-11, ATTRIBUTE-11;
- SYNONYM IS CLASSIFICATION-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:
To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:
- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;

- TKEY part-1, part-2;

VALUES Statement

Purpose:

To specify the allowable range of VALUES, (or specific VALUES), which this SYSTEM-PARAMETER is free to take on. This is useful in determining the need to check data for validity within the system.

Syntax:

```
              {              integer              }
              {                                   }
VALUES ARE    { {  min    }            {   max  } } ;
              { {         }    THRU    {        } }
              { { NEGINF  }            { POSINF } }
```

Complementary Statements:

None.

Usage Rules:

- min and max must be integers
- Each min must be less than the corresponding max.

Synonyms:

VAL VALUE

Examples:

- VALUE 107;
- VALUES ARE 1 THRU 9999;
- VALUE NEGINF THRU POSINF;

4.3 DESIGNATE Section Header Statement

Purpose:

To add additional SYNONYMS to names which already exist within the URL data base. This section is useful in standardizing system names, since one accepted name can be referred to by several different SYNONYM names.

Syntax:

```
DESIGNATE name AS A SYNONYM FOR name

     [ , name AS A SYNONYM FOR name ] ... ;
```

Usage Rules:

-No other statements are allowed in a DESIGNATE section.

-The first name in each pair is taken to be a synonym for the second name in the pair.

Synonyms:

DESG	SYN

Examples:

- DESIGNATE PROC-1 AS A SYNONYM FOR PROCESS-ONE;
- DESIGNATE A-1 AS A SYNONYM FOR ALPHA-MASTER;
- DESG R-1 SYN REPORT-FOR-NEW-MASTER-INPUT;

4.4 ELEMENT Section Header Statement

Purpose:

To allow a detailed description of an ELEMENT. The element is the smallest item of data that can be referred to within the system and still maintain its unique properties.

Syntax:

ELEMENT element-name(s) ;

Usage Rules:

- Must be the first statement in an ELEMENT section.
- Several ELEMENTS may be defined at once.

Synonyms:

ELE ELEMENTS

Examples:

- ELEMENT CHECK-NUMBER;
- ELEMENTS SPAN-NUMBER, SPAN-MILEAGE;
- ELE EMPLOYEE-NUMBER;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ASSOCIATED Statement

Purpose:

To show that the ELEMENT is jointly owned by two ENTITIES which have been described as having a relationship to each other through a RELATION section.

Syntax:

<u>ASSOCIATED</u> WITH relation-name(s) ;

Complementary Statements:

ASSOCIATED-DATA statement in the RELATION section.

Usage Rules:

- Name(s) must be RELATION names.
- An ELEMENT may be associated with several RELATIONS.

Synonyms:

ASOC

Examples:

- ASSOCIATED WITH RELATION-A;
- ASSOCIATED WITH NETWORK-RELATION, DERIVED-RELATION;
- ASOC RELATION-1, RELATION-2;
- ASOC NEW-RELATION;

ATTRIBUTES Statement

Purpose:

To specify properties or characteristics particular to a given section.

Syntax:

```
                                { attv-name } [               { attv-name } ]
ATTRIBUTES ARE attr-name        {           } [  ,attr-name   {           } ]..
                                {  integer  } [               {  integer  } ]
```

Complementary Statements:

none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CLASSIFICATION Statement

Purpose:
To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]
               [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

CLS	CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;
- CLS PING-LEVEL 2, UPDATE;

<u>CONTAINED</u> <u>Statement</u>

Purpose:

To give the GROUPS, ENTITIES, INPUTS, and/or OUTPUTS that contain this ELEMENT. An ELEMENT being contained in a GROUP, ENTITY, INPUT, or OUTPUT means that the data values contained in the ELEMENT will be included in the logical GROUP, ENTITY, INPUT, or OUTPUT.

Syntax:

```
                     group-
                    entity-
      CONTAINED IN   input-name(s)  ;
                    output-
```

Complementary Statements:

CONSISTS statement in the GROUP, ENTITY, INPUT, and OUTPUT sections.

Usage Rules:

-The names must be GROUP, ENTITY, INPUT, or OUTPUT names.

-Several GROUPS, ENTITIES, INPUTS, or OUTPUTS may contain an ELEMENT.

Synonyms:

CNTD

Examples:

- CONTAINED IN GROUP-A1;

- CONTAINED IN ENTITY-1, ENTITY-2;

- CNTD IN INPUT-A;

DERIVED Statement

Purpose:

To give a PROCESS that DERIVES values for the ELEMENT and, optionally, the SETS, INPUTS, ENTITIES, GROUPS, and/or ELEMENTS used in the derivation, and optionally, to specify conditions and/or data associated with the action.

Syntax:

```
                                          [           group-             ]
                                          [          entity-             ]
     DERIVED BY  process-name(s) [ USING       set-     name(s) ]
                                          [           input-             ]
                                          [         element-             ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:

DERIVES or USES statement in a PROCESS section and USED BY statement in a SET, INPUT, ENTITY, GROUP or ELEMENT section. - Several PROCESSES may derive an ELEMENT.

Synonyms:

DRVD USG DPNG DPG EC

Examples:

```
- DERIVED BY PROCESS-A USING INPUT-1;

- DERIVED BY PROCESS-1 USING ENTITY-A, ENTITY-B
          DEPENDING ON CONDITION-A;

- DRVD PROCESS-0 USG INPUT-1
          FOR EACH ELEMENT-B;
```

```
- DRVD PROCESS-NAME USG ENTITY-A, GROUP-B
       DEPENDING ELEMENT-1
       FOR EACH GROUP-1, GROUP-2;
```

ELEMENT SECTION

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

IDENTIFIES Statement

Purpose:
To highlight the fact that this ELEMENT is being used within the system to identify data for storage, retrieval, or processing. This ELEMENT may be considered to be a key.

Syntax:

IDENTIFIES entity-name(s) ;

Complementary Statements:
IDENTIFIED statement in the ENTITY section.

Usage Rules:
-The names must be ENTITY names.

-An ELEMENT may be a potential IDENTIFIER for more than one ENTITY.

Synonyms:

IDS

Examples:

- IDENTIFIES ENT-47;
- IDENTIFIES ENT-784, ENT-6387;
- IDS ENT-957;

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

<u>KEYWORDS</u> ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
    To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

    RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:
    RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- It may be used in any section except the PROBLEM-DEFINER section.

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

    RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
     SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

```
     SM          SEE-MEMOS
```

Examples:

```
     - SEE-MEMO BW-05-03-75-01;

     - SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;

     - SM EPB-37, EPB-38;
```

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SUBSETTING-CRITERION Statement

Purpose:
To indicate that this ELEMENT is used to extract information from a SET to produce a SUBSET.

Syntax:

<u>SUBSETTING-CRITERION</u> FOR set-name(s) ;

Complementary Statements:
SUBSETTING-CRITERIA statement in SET section.

Usage Rules:
- The names must be SET names.
- An ELEMENT may be a SUBSETTING-CRITERION for more than one SET.

Synonyms:

SSCN

Examples:

- SUBSETTING-CRITERION FOR SET-GROUP-BANKS, SET-GROUP-CKTS;
- SSCN: FILE-107, FILE-108;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN SYNONYM

Examples:

- SYNONYMS ARE E-11, ELEMENT-11;
- SYNONYM IS ELEMENT-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

UPDATED Statement

Purpose:

To indicate those PROCESSES which UPDATE this ELEMENT, and optionally, to specify the data used to do the updating, and to express conditions and/or data associated with the action.

Syntax:

```
                                    [          group-              ]
                                    [          entity-             ]
UPDATED BY  process-name(s)  [ USING    element-      name(s) ]
                                    [          input-              ]
                                    [            set-              ]

    [DEPENDING ON element-      name(s) ]
    [             condition-            ]

    [             group-                ]
    [             entity-               ]
    [FOR EACH     element-      name(s) ];
    [             output-               ]
    [             input-                ]
    [             set-                  ]
```

Complementary Statements:

UPDATES or USES statement in PROCESS section and USED BY statement in INPUT, SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

-An ELEMENT may be updated by more than one PROCESS.

Synonyms:

UPPD USG DPNG DPG EC

Examples:

- UPDATED BY P-101 FOR EACH INPUT-A, INPUT-B;

```
- UPDD P-103, OUTPUT-P-675354 USING MASTER-FILE-6
        DEPENDING CONDITION-A
        FOR EACH SET-B, SET-C;
```

USED Statement

Purpose:

To indicate the PROCESS(ES) that USE(D) this ELEMENT, and optionally, DERIVE(S) OUTPUTS or UPDATE(S) SETS, ENTITIES, GROUPS, and/or ELEMENTS, and to specify conditions and/or iterations associated with DERIVE(s) or UPDATE(s).

Syntax:

```
              set-           [                        set-              ]
            input-           [      { DERIVE }    *output-              ]
   USES   element-name(s)    [ TO   {        }    element-     name(s)  ]
            group-           [      { UPDATE }      group-              ]
           entity-           [                     entity-              ]

        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:

USES, UPDATES or DERIVES statement in a PROCESS section and DERIVED or UPDATED statement in SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

- Several PROCESSES may use the ELEMENT.
- DEPENDING or FOR EACH statements can only be used with the DERIVE or UPDATE clauses.

Synonyms:

DRV UPD DPNG DPG EC

Examples:

```
     - USED BY PROCESS-UPDATE;
     - USED BY LINEAR-PROCESS, INTEGER-PROCESS TO DERIVE ALPHA
                DPG LINEAR-FUNCTION
                EC INPUT-FUNCTION;
```

<u>VALUES Statement</u>

Purpose:
To specify the allowable range of VALUES, or specific VALUES, which this ELEMENT is free to take on. This is useful in determining the need to check data for validity within the system.

Syntax:

```
                {              integer                   }
                {                                        }
     VALUES ARE { { min    }            {    max   } }  ;
                { {        }    THRU    {          } }
                { { NEGINF }            { POSINF   } }
```

Complementary Statements:
None.

Usage Rules:
- min and max must be integers
- Each min must be less than the corresponding max.

Synonyms:

VAL	VALUE

Examples:

- VALUE 107;
- VALUES ARE 1 THRU 9999;
- VALUE NEGINF THRU POSINF;

4.5 ENTITY Section Header Statement

Purpose:

To allow a detailed description of the contents of an ENTITY. An ENTITY is a logical, usable collection of data that serves a unique purpose within the system. An ENTITY is information used by the target system that represents an object or concept of the real world. It is required by the target system for information processing purposes.

Syntax:

```
ENTITY entity-name(s) ;
```

Usage Rules:

-It must be the first statement in an ENTITY section.

-Several ENTITIES may be defined at once.

Synonyms:

ENT   ENTITIES

Examples:

- ENTITY ROOT-SEGMENT;
- ENTITY NH-SEGMENT, NI-SEGMENT;
- ENT ENTITY-1;
- ENT NS-SEGMENT, NF-SEGMENT;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
    ASSERT name attribute-name attribute-value
                        [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

- ASSERT data-name-1 type character;
- ASRT sine-function arguments 1,
  coord-function arguments 2;

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                               { attv-name } [                  { attv-name }  ]
      ATTRIBUTES ARE attr-name {           } [  ,attr-name      {             }  ] ..
                               {  integer  } [                  {  integer    }  ]
```

Complementary Statements:
none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CARDINALITY Statement

Purpose:

To define the number of times this ENTITY appears in the system. This can be used to estimate the size of SETS that contain the ENTITY.

Syntax:

CARDINALITY IS system-parameter ;

Complementary Statements:

None.

Usage Rules:

-An ENTITY may only have one CARDINALITY.

Synonyms:

CARD OCCS OCCURRENCES

Examples:

- CARDINALITY IS ONE;

- CARD ONE;

CLASSIFICATION Statement

Purpose:

To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]

              [, classification-name [ integer ]]... ;
```

Complementary Statements:

None.

Usage Rules:

- The name must be a CLASSIFICATION name.

Synonyms:

CLS	CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;

- CLS RING-LEVEL 2, UPDATE;

CONSISTS Statement

Purpose:

To describe the combination of GROUPS and/or ELEMENTS which make up this ENTITY. This implies that each instance of the ENTITY will contain values of the GROUP and ELEMENT names. A GROUP or ELEMENT may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
                                          element-
    CONSISTS OF [ system-parameter ]   group-name

                                                element-
                  [ , [ system-parameter ]   group-name ] ... ;
```

Complementary Statements:

CONTAINED statement in the GROUP and ELEMENT sections.

Usage Rules:

- The names, other than the SYSTEM-PARAMETERS , must be GROUP or ELEMENT names.

- An ENTITY can contain several GROUPS or ELEMENTS.

Synonyms:

CSTS

Examples:

- CONSISTS OF ONE GR-1, ONE GR-2, TWO ELE-5 ;

- CONSISTS OF: UNIQUE-SPAN-NUMBER;

- CSTS TWO ELE-A, GROUP-7 ;

CONTAINED Statement

Purpose:
To give the SETS that contain this ENTITY. An ENTITY being contained in a SET means that the data values contained in the ENTITY will be included in the logical SET.

Syntax:

CONTAINED IN set-name(s) :

Complementary Statements:
CONSISTS statement in a SET section.

Usage Rules:
-The names must be SET names.

-An ENTITY can be contained in several SETS.

Synonyms:

CNTD

Examples:

```
- CONTAINED IN INPUT-HS;

- CONTAINED IN: HS-1, HS-2, HS-3;

- CNTD IN FIRST-HS;

- CNTD: HS-ONE, OUTPUT-HS-ONE;

- CNTD: MASTER-FILE;

- CONTAINED PAYROLL-CHANGE, NAME-DELETE;

- CNTD NEW-EMPLOYEE;
```

DERIVED Statement

Purpose:
To give a PROCESS that DERIVES values for the ENTITY and, optionally, the SETS, INPUTS, ENTITIES, GROUPS, and/or ELEMENTS used in the derivation, and to specify conditions and/or iterations associated with the action.

Syntax:

```
                                  [              group-          ]
                                  [             entity-          ]
     DERIVED BY  process-name(s)  [  USING         set-   name(s)]
                                  [              input-          ]
                                  [            element-          ]

          [DEPENDING ON element-     name(s) ]
          [             condition-           ]

          [             group-               ]
          [             entity-              ]
          [FOR EACH     element-     name(s) ];
          [             output-              ]
          [             input-               ]
          [             set-                 ]
```

Complementary Statements:
DERIVES or USES statement in a PROCESS section and USED BY statement in a SET, INPUT, ENTITY, GROUP or ELEMENT section.

Usage Rules:
-Several PROCESSES may derive an ENTITY.

Synonyms:

DRVD USG DPNG DPG EC

Examples:

```
- DERIVED BY A-PROCESS USING FILE-1
          DPNG CONDITION-A;
```

```
- DERIVED R-PROCESS USING ENTITY-456
        FOR EC SET-A;

- DRVD OUT-PROCESS USG GROUP-SPAN-13
        DRNG CONDITION-A
        FOR EC SET-A;
```

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

IDENTIFIED Statement

Purpose:

To give the possible GROUPS and/or ELEMENTS which identify this ENTITY. This is necessary to uniquely distinguish multiple instances of the same ENTITY. This statement can be viewed as defining a unique key for information retrieval purposes.

Syntax:

```
                     group
     IDENTIFIED BY element-name(s) ;
```

Complementary Statements:

IDENTIFIES statements in GROUP and ELEMENT sections.

Usage Rules:

-The names must be either GROUP or ELEMENT names.

-An ENTITY may have several alternative identifiers.

-If the ENTITY is IDENTIFIED by a GROUP then the ELEMENTS which make up the GROUP are taken together as an identifier.

Synonyms:

IDD

Examples:

- IDENTIFIED BY SPAN-NUMBER;

- IDENTIFIED BY SPAN-NUMBER, SPAN-LOG;

- IDD ELEMENT-1, GROUP-1;

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
     KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RELATED Statement

Purpose:
To identify which RELATIONS and ENTITIES this ENTITY is associated with.

Syntax:

```
RELATED TO entity-name VIA relation-name ;
```

Complementary Statements:
BETWEEN statement in the RELATION section.

Usage Rules:
- The second name must be a RELATION name.
- The first name must be an ENTITY name.
- All RELATIONS are binary.

Synonyms:

REL

Examples:

```
- RELATED TO NH-ENTITY VIA UPDATE-RELATION;

- REL NI-SEG VIA NI-RELATION;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:
RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

<u>SECURITY</u> IS security-name(s) ;

Complementary Statements:
APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

```
-SOURCES SPP-3-0;

-SRC ENG-LETTER-1-MAY-1973;
```

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

```
- SEE-MEMO BW-05-03-75-01;

- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;

- SM EPB-37, EPB-38;
```

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-9;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a MEMO section, or a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE E-11, ENTITY-11;

- SYNONYM IS ENTITY-11;

- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;

- TKEY part-1, part-2;

UPDATED Statement

Purpose:
To indicate those PROCESSES which update this ENTITY, and optionally, to specify the data used to do the updating, and to specify conditions and/or iterations associated with the action.

Syntax:

```
                                          [        group-            ]
                                          [       entity-            ]
     UPDATED BY  process-name(s) [ USING  element-    name(s) ]
                                          [        input-            ]
                                          [          set-            ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:
UPDATES or USES statement in PROCESS section and USED BY statement in INPUT, SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:
-An ENTITY may be UPDATED by more than one PROCESS.

Synonyms:

UPDD USG DPNG DPS EC

Examples:

```
    - UPDATED BY P-101;

    - UPDD P-103, OUTPUT-P-675354 USING MASTER-FILE-4
          DPNG ON CONDITION-A
          FOR EC ELEMENT-B;
```

USED Statement

Purpose:

To indicate the PROCESS(ES) that USE(D) this ENTITY, and optionally, DERIVE(S) OUTPUTS or UPDATE(S) SETS, ENTITIES, GROUPS, and/or ELEMENTS, and to specify conditions and/or iterations associated with the DERIVE or UPDATE statements.

Syntax:

```
                                [                  set-               ]
                                [    { DERIVE }  *output-             ]
USED BY process-name(s)  [  TO  {          }   entity-      name(s) ]
                                [    { UPDATE }   group-              ]
                                [                 element-            ]

      [DEPENDING ON element-     name(s) ]
      [             condition-           ]

      [             group-               ]
      [             entity-              ]
      [FOR EACH     element-     name(s) ];
      [             output-              ]
      [             input-               ]
      [             set-                 ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:

USES, UPDATES or DERIVES statement in a PROCESS section and DERIVED or UPDATED statement in SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

- Several PROCESSES may use the ENTITY.
- DEPENDING ON or FOR EACH statements can only be used with DERIVE or UPDATE clauses.

Synonyms:

DRV UPD DPNG DPG FC

Examples:

```
    - USED BY PROCESS;

    - USED BY LINEAR-PROCESS, INTEGER-PROCESS TO UPDATE ENT-1
              DPNG LINEAR-FUNCTION
              FOR EC INPUT-FUNCTION;
```

VOLATILITY Statement

Purpose:
    To give a measure of the changability of the ENTITY.

Syntax:

```
    VOLATILITY ;
           comment-entry ;
```

Complementary Statements:
    None.

Usage Rules:
    -Only one VOLATILITY statement may be given for an ENTITY.

Synonyms:

    VOL

Examples:

```
    VOLATILITY;

        SEGMENT IS UPDATED EACH TIME AN SR TRANSACTION IS REQUESTED;
```

4.6 EVENT Section Header Statement

Purpose:

To describe the dynamic occurrences which take place within the target system. An EVENT is used to describe an instance of time during the operation of the target system. An EVENT may re-occur more than once during target system operation. For example, " occurrence of error " may be an EVENT which causes normal processing to be suspended while an error processor is initiated. An EVENT may occur when a PROCESS is started or finished, when a CONDITION becomes TRUE or FALSE, when an INPUT becomes available, or when another EVENT occurs.

Syntax:

<u>EVENT</u> event-name(s) ;

Usage Rules:

-It must be the first statement in an EVENT section.

-Several EVENTS may be defined at once.

Synonyms:

EV	EVT	EVENTS

Examples:

- EVENT TIME-CARD-ENTRY;
- EVENTS REGISTER, CHECK-IN, CHECK-OUT;
- EV CARRIER-ALARM;
- EVT CARRIER-ALARM, CARRIER-FAILURE;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value

            [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

<u>ATTRIBUTES Statement</u>

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                          { attv-name } [             { attv-name } ]
ATTRIBUTES ARE attr-name  {           } [ ,attr-name  {           } ]..
                          {  integer  } [             {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CAUSED Statement

Purpose:

To specify INPUT(S), CONDITION(S), or additional EVENT(S) which cause this EVENT, and optionally, to specify conditions and/or data associated with the action.

Syntax:

```
                    event-
     CAUSED BY              name(s)
                    input-

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]


                                            { TRUE  }
     CAUSED WHEN condition-name BECOMES     {       };
                                            { FALSE }
```

Complementary Statements:

CAUSES statement in the EVENT and INPUT sections, and BECOMING CAUSES statement in the CONDITION section.

Usage Rules:

- AN EVENT may be CAUSED by any number of EVENTS and/or INPUTS.

- A separate statement is required for each CONDITION change which CAUSES an EVENT. Any number of such statements may appear in a single EVENT section.

Synonyms:

CSD DPNG DPG EC

Examples:

```
     - CAUSED BY TIME-CARD-INPUT, DEADLINE-REACHED;

     - CAUSED WHEN ERROR-FLAG-SET BECOMES TRUE;

     - CSD ORDERS DRNG STOCK-READY FC CUSTOMER;
```

CAUSES Statement

Purpose:
To specify other EVENT(S) which are caused by this EVENT, and to specify, optionally, conditions and/or data associated with the action.

Syntax:

```
     CAUSES event-name(s)

          [DEPENDING ON element-     name(s) ]
          [             condition-           ]

          [             group-               ]
          [             entity-              ]
          [FOR EACH     element-     name(s) ];
          [             output-              ]
          [             input-               ]
          [             set-                 ]
```

Complementary Statements:
CAUSED statement in the EVENT section.

Usage Rules:
- An EVENT may CAUSE several other EVENTS.

Synonyms:

CSS DPNG DPG EC

Examples:

```
- CAUSES SUBPROCESS-COMPLETION, MAIN-PROCESS-COMPLETION ;

- CSS ERROR-DETECTED
          DPG TIME-CARD-FOUND
          EC EMPLOYEE-RECORD;
```

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
    DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

HAPPENS Statement

Purpose:

One purpose is to give the number of times an EVENT occurs during an INTERVAL. More than one instance of an EVENT may occur over some period of time. The number of instances of the EVENT which occur in a time INTERVAL is expressed with this statement. Another purpose is to declare that the EVENT occurs repetitively in a specific cycle. Lastly, this statement may be used to specify that the EVENT occurs after some delay, or at a particular time.

Syntax:

```
                {system-parameter TIMES-PER interval-name}
        HAPPENS {EVERY system-parameter interval-name     };
                {[WITHIN] system-parameter interval-name  }
                {                AFTER event-name         }
```

Complementary Statements:

None.

Usage Rules:

- The statement may be given as many times as necessary for different INTERVALS.

- Combination of HAPPENS statements cannot be used for same INTERVAL name.

Synonyms:

HAP TIMP EVR EVY WI WIN WTH AF

Example:

- HAPPENS FORTY-SEVEN TIMES-PER INTERVAL-A;

- HAP THIRTY-TWO TIMP INT-B;

- HAP EVY ONE MONTH;

```
- HAP ONE DAY AF EVENT-A;

- HAP WTH TWO WEEKS AF EVENT-B;
```

INCEPTION Statement

Purpose:

To specify those PROCESS(ES) whose inception causes this EVENT.

Syntax:

ON INCEPTION OF process-name(s) ;

Complementary Statements:

INCEPTION-CAUSES statement in a PROCESS section.

Usage Rules:

-The names must be PROCESS names.

-Several PROCESSES may be given.

Synonyms:

INCP

Examples:

- ON INCEPTION OF PROCESS-IN;

- INCEPTION OF PROCESS-OUT;

- INCP SORT-ALPHA;

INTERRUPTS Statement

Purpose:

To specify those PROCESS(ES) which are interrupted as a result of this EVENT, and optionally, to specify conditions and/or interations associated with the action.

Syntax:

```
     INTERRUPTS process-name(s)

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:

INTERRUPTED statement in the PROCESS section.

Usage Rules:

- An EVENT may INTERRUPT several PROCESSES.

Synonyms:

INTS DPNG DPG EC

Examples:

```
     - INTERRUPTS MAIN-PROCESSING ;

     - INTS MASTER-FILE-SEARCH, PAYSYSTEM-PROCESSING
            DPG TIME-CARD
            FOR EC EMPLOYEE-LIST;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

<u>KEYWORDS</u> ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a KEYWORD.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS COM-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MAKES Statement

Purpose:

To give those CONDITION(S) which are set by this EVENT, a
optionally, to specify conditiors and/or iterations associ
with the action.

Syntax:

```
MAKES condition-name(s) { TRUE  }
                        {       }
                        { FALSE }
     [DEPENDING ON element-       name(s) ]
     [             condition-             ]
     [                                    ]
     [             group-                 ]
     [FOR EACH     entity-                ]
     [             element-       name(s) ];
     [             output-                ]
     [             input-                 ]
                   set-                   ]
                                          ]
```

Complementary statements:
MADE statement in the CONDITION section.

Usage Rules:

- An EVENT may make several CONDITIONS become TRUE or FALSE.
- An EVENT cannot MAKE some CONDITION(S) TRUE and other CONDITION(S) FALSE in the same statement. Separate statements are required.

Synonyms:

MAK DPNG DPG EC

Examples:

- MAKES PROCESS-COMPLETION TRUE ;
- MAK ERROR-OCCURRENCE, OUTPUT-INTERRUPTION F

MAKES Statement

Purpose:

To give those CONDITION(S) which are set by this EVENT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
                                   { TRUE  }
     MAKES condition-name(s)       {       }
                                   { FALSE }

          [DEPENDING ON element-     name(s) ]
          [             condition-           ]

          [             group-               ]
          [             entity-              ]
          [FOR EACH     element-     name(s) ];
          [             output-              ]
          [             input-               ]
          [             set-                 ]
```

Complementary Statements:

MADE statement in the CONDITION section.

Usage Rules:

- An EVENT may make several CONDITIONS become TRUE or FALSE.
- An EVENT cannot MAKE some CONDITION(S) TRUE and other CONDITION(S) FALSE in the same statement. Separate statements are required.

Synonyms:

MAK DPNG DPG EC

Examples:

- MAKES PROCESS-COMPLETION TRUE ;
- MAK ERROR-OCCURRENCE, OUTPUT-INTERRUPTION F

```
DRNG ELEMENT-A
EC INPUT-B;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
    SECURITY IS security-name(s) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

```
    SEC          SECURITIES
```

Examples:

```
    - SECURITY IS PROJECT-MANAGER;

    - SECURITIES ARE D-ORMISTON, S-MENNEL;

    - SEC L-HANNON;
```

SEE-MEMO Statement

Purpose:
To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:
APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

```
SYNONYMS ARE synonym-name(s) ;
```

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE E-11, EVENT-11;
- SYNONYM IS EVENT-11;
- SYN ALPHA;

TERMINATES Statement

Purpose:

To specify a PROCESS/PROCESSES that are terminated by this EVENT, and optionally, to specify conditions and/or iteratins associated with the action.

Syntax:

```
TERMINATES process-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:

TERMINATED statement in PROCESS section.

Usage Rules:

- An EVENT may TERMINATE several PROCESSES.

Synonyms:

TRMS DPNG DPG FC

Examples:

```
- TERMINATES INPUT-PROCESSING;

- TRMS PROC-A, PROC-B, PROC-C
           DPNG CONDITION-1, CONDITION-2
           FOR EACH INPUT-A;
```

TERMINATION Statement

Purpose:
To indicate those PROCESS(ES) on whose TERMINATION this EVENT occurs.

Syntax:

```
ON TERMINATION OF process-name(s) ;
```

Complementary Statements:
TERMINATION-CAUSES statement in a PROCESS section.

Usage Rules:
- The names must be PROCESS names.
- Several PROCESSES may be given.

Synonyms:

TERM

Examples:

```
- ON TERMINATION OF INPUT-PROCESS;

- TERMINATION UPDATE-PROCESS;

- TERM FORECAST-PROCESS;
```

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

TRIGGERS Statement

Purpose:
To give the PROCESS/PROCESSES which are triggered when this EVENT occurs, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
TRIGGERS process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]
```

Complementary Statements:
TRIGGERED statement in PROCESS section.

Usage Rules:
- The names must be PROCESS names.
- Several PROCESSES may be triggered by any EVENT.

Synonyms:

TRGS DPNGH DPG EC

Examples:

```
    - TRIGGERS UPDATE-PROCESS;

    - TRIGGERS P-101,P-420,P-7598
               FOR EC INPUT-A;

    - TRGS EXTRA-LINK-PROCESS
               DPG CONDITION-A
               FOR EC INPUT-B;
```

4.7 GROUP Section Header Statement

Purpose:

To allow a detailed description of a GROUP. A GROUP is a logical collection of data ELEMENTS and/or other GROUPS. A GROUP is a collection of information which can be CONTAINED in larger collections of information. E.g. INPUTS, OUTPUTS, and ENTITIES. For instance, current-date might be a GROUP containing month, day and year.

Syntax:

GROUP group-name(s) ;

Usage Rules:

-It must be the first statement in a GROUP section.

-Several GROUPS may be defined at once.

Synonyms:

GR	GROUPS

Examples:

- GROUP SPAN-MAKEUP;
- GROUPS: SPAN-A, LINK-A;
- GR GROUP-A;
- GR: SPAN-784, LINK-737;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
        ASSERT name attribute-name attribute-value

                        [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ASSOCIATED Statement

Purpose:

To show that the GROUP is jointly owned by two ENTITIES which have been described as having a relationship to each other through a RELATION section.

Syntax:

<u>ASSOCIATED</u> WITH relation-name(s) ;

Complementary Statements:

ASSOCIATED-DATA statement in a RELATION section.

Usage Rules:

-The names must be RELATION names.

-A GROUP may be associated with several RELATIONS.

Synonyms:

ASOC

Examples:

- ASSOCIATED WITH EMPLOYED-BY-RELATION;
- ASSOCIATED WITH NAME-RELATION, DATE-RELATION, TIME-RELATION;
- ASOC RELATION-C1;
- ASOC RELATION-C1,RELATION-C2,RELATION-C3;

ATTRIBUTES Statement

Purpose:

To specify properties or characteristics particular to a given section.

Syntax:

```
                              { attv-name } [                 { attv-name }  ]
     ATTRIBUTES ARE attr-name {             } [ ,attr-name {               }  ] ..
                              {  integer  } [                 {  integer    }  ]
```

Complementary Statements:

none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CLASSIFICATION Statement

Purpose:
To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]

                    [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

CLS	CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;

- CLS RING-LEVEL 2, UPDATE;

CONSISTS Statement

Purpose:

To describe the combination of other GROUPS and/or ELEMENTS which make up this GROUP. This implies that each instance of the GROUP will contain values of the GROUP and ELEMENT names. A GROUP or ELEMENT may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
                                             element-
     CONSISTS OF [ system-parameter ]   group-name

                                                  element-
                 [ , [ system-parameter ]   group-name ] ... ;
```

Complementary Statements:

CONTAINED statement in a GROUP or ENTITY section.

Usage Rules:

- The names, other than the system-parameters, must be GROUP or ELEMENT names.
- A GROUP can contain several GROUPS or ELEMENTS.

Synonyms:

CSTS

Examples:

- CONSISTS OF TWO DATA-GROUP-1;
- CONSISTS: DATA-GROUP-1, ELEMENT-A;
- CSTS OF SPAN-ELEMENT-A;
- CSTS: GROUP-NO-1, GROUP-NO-2;

CONTAINED Statement

Purpose:

To give the ENTITIES, INPUTS, OUTPUTS, or GROUPS that contain this GROUP. A GROUP being contained in a GROUP, ENTITY, INPUT, or OUTPUT means that the data values contained in the GROUP will be included in the logical GROUP, ENTITY, INPUT, or OUTPUT.

Syntax:

```
                      group-
                      entity-
CONTAINED IN          input-    name(s) ;
                      output-
```

Complementary Statements:

CONSISTS statement in GROUP, ENTITY, INPUT and OUTPUT sections .

Usage Rules:

-The names must be GROUP, ENTITY, INPUT or OUTPUT names.

-A GROUP may be contained in several GROUPS, ENTITIES, INPUTS or OUTPUTS.

Synonyms:

CNTD

Examples:

- CONTAINED IN GROUP-1;

- CONTAINED IN GROUP-2, INPUT-2, OUTPUT-REP;

- CNTD IN FIRST-ENTITY;

DERIVED Statement

Purpose:

To give a PROCESS that DERIVES values for the GROUP and, optionally, the SETS, INPUTS, ENTITIES, GROUPS, and/or ELEMENTS used in the derivation, and to specify conditions and/or iterations associated with the action.

Syntax:

```
                                    [         group-              ]
                                    [        entity-              ]
     DERIVED BY  process-name(s)    [ USING     set-    name(s)   ]
                                    [         input-              ]
                                    [       element-              ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:

DERIVES or USES statement in a PROCESS section and USED BY statement in a SET, INPUT, ENTITY, GROUP or ELEMENT section.

Usage Rules:

- Several PROCESSES may derive a GROUP.

Synonyms:

DRVD USG DPNG DPG EC

Examples:

- DERIVED BY PROC-NAME USING GROUP-22;

- DERIVED BY PAYROLL-PROCESSING USING PAY-MAST, PAY-STMT;

- DRVD SPAN-UPDATE USG SPAN-NO, MILES
                    GROUP SECTION

```
DPNG CONDITION-A
FOR EC ELEMENT-A, ELEMENT-B;
```

<u>DESCRIPTION Statement</u>

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

IDENTIFIES Statement

Purpose:

To highlight the fact that this GROUP is being used within the system to identify data for storage, retrieval, or processing. This GROUP may be considered to be a key in the target system.

Syntax:

<u>IDENTIFIES</u> entity-name(s) ;

Complementary Statements:

IDENTIFIED statement in ENTITY section.

Usage Rules:

- The names must be ENTITY names.
- A GROUP may IDENTIFY several different ENTITIES.
- If an ENTITY is identified by a GROUP, then the ELEMENTS which make up the GROUP taken together form the identifier.

Synonyms:

IDS

Examples:

- IDENTIFIES ENTITY-743;
- IDENTIFIES ENTITY-78954, ENTITY-8;
- IDS ENT-3;

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

KEYWORDS ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

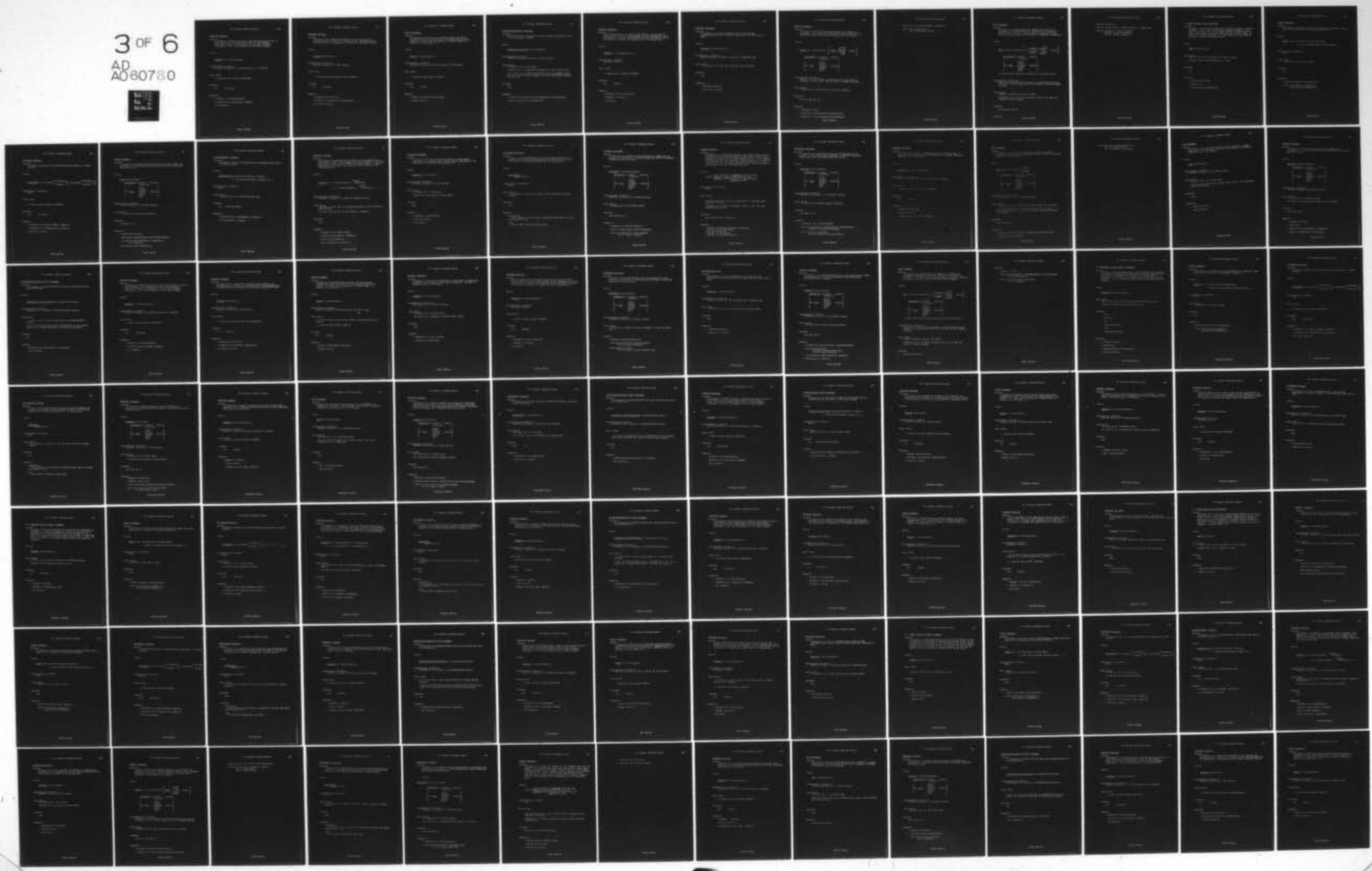
RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

AD-A060 780 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78 F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-130-VOL-2 NL

3 OF 6
AD
A060780



SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
SECURITY IS security-name(s) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

```
- SEE-MEMO BW-05-03-75-01;

- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;

- SM EPB-37, EPB-38;
```

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SUBSETTING-CRITERION Statement

Purpose:
To indicate that this GROUP is used to extract information from a SET to produce a SUBSET.

Syntax:

SUBSETTING-CRITERION FOR set-name(s) ;

Complementary Statements:
SUBSETTING-CRITERIA statement in SET section.

Usage Rules:
-The names must be SET names.

-A GROUP may be a SUBSETTING-CRITERION for more than one SET.

-If a GROUP is a SUBSETTING-CRITERION then the ELEMENTS which make up the GROUP taken together form the SUBSETTING-CRITERION for that SET.

Synonyms:

SSCN

Examples:

- SUBSETTING-CRITERION FOR HS-GROUP-BANKS, HS-GROUP-CKTS;

- SSCN: HS-GROUP-107, HS-GROUP-108;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE G-11, GROUP-11;
- SYNONYM IS GROUP-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

UPDATED Statement

Purpose:
To indicate those PROCESSES which update this GROUP, and optionally, to specify the data used to do the updating, and conditions and/or iterations associated with the action.

Syntax:

```
                                      [         group-             ]
                                      [        entity-             ]
     UPDATED BY   process-name(s)  [ USING  element-    name(s) ]
                                      [         input-             ]
                                      [           set-             ]

        [DEPENDING ON element-      name(s) ]
        [             condition-            ]

        [             group-                ]
        [             entity-               ]
        [FOR EACH     element-      name(s) ];
        [             output-               ]
        [             input-                ]
        [             set-                  ]
```

Complementary Statements:
UPDATES or USES statement in PROCESS section and USED BY statement in INPUT, SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:
-A GROUP may be UPDATED by more than one PROCESS.

Synonyms:

UPDD USG DPNG DPG EC

Examples:

- UPDATED BY P-101;

- UPDD P-103, OUTPUT-P-675354 USING FILE-A;

- UPDD P-105, P-107 DEPENDING ON CONDITION-A;

```
- UPDD P-109, FOR EACH ELEMENT-A, ELEMENT-B;

- UPDD P-111, USING FILE-B.
       DPNG CONDITION-B INPUT-A;
```

GROUP SECTION

USED Statement

Purpose:

To indicate the PROCESS(ES) that USE(D) this GROUP, and optionally, DERIVE(S) OUTPUTS or UPDATE(S) SETS, ENTITIES, GROUPS, and/or ELEMENTS, and to specify conditions and/or iterantions associated with DERIVE(s) or UPDATE(s) statements.

Syntax:

```
                               [                       set-                 ]
                               [    { DERIVE }  *output-                    ]
USED BY process-name(s)        [ TO {        }   entity-       name(s)      ]
                               [    { UPDATE }   group-                     ]
                               [               element-                     ]

          [DEPENDING ON element-     name(s) ]
          [             condition-           ]

          [             group-               ]
          [             entity-              ]
          [FOR EACH     element-     name(s) ];
          [             output-              ]
          [             input-               ]
          [             set-                 ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:

USES, UPDATES or DERIVES statement in a PROCESS section and DERIVED or UPDATED statement in SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

-Several PROCESSES may use the GROUP.

-DEPENDING ON or FOR EACH statements can only be used with DERIVE or UPDATE clauses.

Synonyms:

DRV UPD DPNG DPG EC

Examples:

GROUP SECTION

```
- USED BY PROCESS-A;

- USED BY LINEAR-PROCESS, INTEGER-PROCESS TO UPDATE GR-4;

- USED BY PROCESS-B TO DERIVE OUTPUT-1
      DEPENDING ON ERROR-OCCURRENCE
      FOR EACH INPUT-1, INPUT-2;
```

4.8 INPUT Section Header Statement

Purpose:
To allow a detailed description of an INPUT. An INPUT is used to describe a collection of information produced external to the target system but used by the target system. An INPUT shows the flow of data from the outside world into the system. Hence, it crosses the system boundary. The INPUT section is also used to uniquely identify each system input.

Syntax:

<u>INPUT</u> input-name(s) ;

Usage Rules:
-Must be the first statement in a INPUT section.

-Several INPUTS may be defined at a time.

Synonyms:

INP

Examples:

- INPUT PAYROLL-CODE;

- INPUT CODE ;

- INP DATA-FOR-COMMUNICATION;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                  [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTE Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                                 { attv-name } [                { attv-name } ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name {             } ]...
                                 {  integer  } [                {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZ79V9;

CAUSES Statement

Purpose:
To specify an EVENT/EVENTS which are caused by this INPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
CAUSES event-name(s)
        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:
CAUSED statement in the EVENT section.

Usage Rules:
- An INPUT may CAUSE several EVENTS.

Synonyms:

CSS DPNG DPG EC

Examples:

```
- CAUSES START-PROC-A;

- CSS SUBPROCESS-COMPLETION, MAIN-PROCESS-BEGINS ;

- CSS EVENT-A DBNG CONDITION-A, CONDITION-B
        FOR EC GROUP-1;

- CSS EVENT-B DPNG CONDITION-C;
```

CLASSIFICATION Statement

Purpose:

To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]
                 [, classification-name [ integer ]]... ;
```

Complementary Statements:

None.

Usage Rules:

- The name must be a CLASSIFICATION name.

Synonyms:

CLS   CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;
- CLS RING-LEVEL 2, UPDATE;

CONSISTS Statement

Purpose:

To describe the combination of GROUPS, and/or ELEMENTS which make up this INPUT. This implies that each instance of the INPUT will contain values of the GROUP and ELEMENT names. A GROUP or ELEMENT may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
                                          element-
CONSISTS OF [ system-parameter ]   group-name
                                                    element-
              [ , [ system-parameter ]   group-name ] ... ;
```

Complementary Statements:

CONTAINED statement in a GROUP or ELEMENT section.

Usage Rules:

- The names, other than the system-parameters, must be GROUP or ELEMENT names.
- An INPUT can contain several GROUPS or ELEMENTS.

Synonyms:

CSTS

Examples:

- CONSISTS OF TWO DATA-GROUP-1;
- CONSISTS: DATA-GROUP-1, ELEMENT-A;
- CSTS OF SPAN-ELEMENT-A;
- CSTS: GROUP-NO-1, GROUP-NO-2;

CONTAINED Statement

Purpose:
To give the SETS that contain this INPUT. An INPUT being contained in a SET means that the data values contained in the INPUT will be included in the logical SET.

Syntax:

```
CONTAINED IN set-name(s) ;
```

Complementary Statements:
CONSISTS statement in an SET section.

Usage Rules:
- The names must be SET names.
- Several SETS may contain a given INPUT.

Synonyms:

CNTD

Examples:

- CONTAINED IN MASTER-FILE;
- CNTD: HS-1,HS-2;
- CNTD FILE-1;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

GENERATED Statement

Purpose:
To identify the INTERFACE which produces this INPUT for the system, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
GENERATED BY interface-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:
GENERATES statement in INTERFACE section.

Usage Rules:
-The names must be INTERFACE names.

Synonyms:

GEND DPNG DPG EC

Examples:

- GENERATED BY INPUT-INTERFACE-1;

- GEND BY INTERFACE-456 DPNG ON ELEMENT-A;

- GEND BY INTERFACE-500 DPNG ELEMENT-B
  FOR EC INPUT-A, INPUT-B;

HAPPENS Statement

Purpose:

One purpose is to give the volume of this INPUT. More than one instance of an INPUT may occur over some period of time. The number of instances of the INPUT which occur in a time INTERVAL is expressed with this statement. Another purpose is to declare that instances of the INPUT occur repetitively in a specific cycle. Lastly, this statement may be used to specify a delay or a particular time that the INPUT may occur.

Syntax:

```
         {system-parameter TIMES-PER interval-name}
HAPPENS  {EVERY system-parameter interval-name     };
         {[WITHIN] system-parameter interval-name  }
         {                 AFTER event-name        }
```

Complementary Statements:

None.

Usage Rules:

-The statement may be given as many times as necessary with different INTERVAL

-Combination of HAPPENS statements cannot be used for same INTERVAL name. Names.

Synonyms:

HAP TIMP EVR EVY WI WTN WTH AF

Examples:

- HAPPENS FORTY-SEVEN TIMES-PER INTERVAL-A;
- HAP THIRTY-TWO TIMP INT-B;
- HAP EVY ONE MONTH;
- HAP ONE DAY AF EVENT-A;
- HAP WTN TWO WEEKS AF EVENT-B;

INTERRUPTS Statement

Purpose:

To specify those PROCESS(ES) which are interrupted by the arrival of this INPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
INTERRUPTS process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

INTERRUPTED statement in the PROCESS section.

Usage Rules:

- An INPUT may INTERRUPT several PROCESSES.

Synonyms:

INTS DPNG DPG EC

Examples:

```
- INTERRUPTS PAYCHECK-PROCESSING;

- INTS LOADING-PROC-A, LOADING-PROC-B, LOADING-PROC-C
          DEPENDING ON ERROR-OCCURRENCE;

- INTS PROCESS-A, PROCESS=B
          DPNG CONDITION-100 FOR EC GROUP-A;
```

KEYWORDS Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for a keyword.

Usage Rules:

- A section may have several KEYWORDS.

Synonyms:

KEY KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MAKES Statement

Purpose:
To give those CONDITION(S) which are set when this INPUT arrives, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
                                 { TRUE  }
    MAKES condition-name(s)      {       }
                                 { FALSE }

         [DEPENDING ON element-    name(s) ]
         [             condition-          ]

         [             group-              ]
         [             entity-             ]
         [FOR EACH     element-    name(s) ];
         [             output-             ]
         [             input-              ]
         [             set-                ]
```

Complementary Statements:
MADE statement in the CONDITION section.

Usage Rules:
- An INPUT may make several CONDITIONS become TRUE or FALSE.

- An INPUT cannot MAKE some CONDITION(S) TRUE and some CONDITION(S) FALSE in a single statement. Separate statements are required.

Synonyms:

MAK DPNG DPG FC

Examples:

- MAKES END-OF-FILE-REACHED, INPUT-PROC-COMPLETION TRUE ;

- MAK SYSTEM-READY FALSE;

```
- MAK FATAL-ERROR, PROGRAM-INTERRUPT T
      DPG ERROR-OCCURRENCE
      FOR EC ELEMENT-A, ELEMENT-B;
```

PART Statement

Purpose:

To show the structural relationship of this INPUT to a higher-level INPUT. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

PART OF input-name ;

Complementary Statements:

SUBPARTS statement in an INPUT section.

Usage Rules:

- The name must be an INPUT name.
- Only one INPUT name may be given, hence, only a tree structure can be established.

Synonyms:

none.

Examples:

- PART OF IN-101;
- PART INPUT-35;

RECEIVED Statement

Purpose:

To show which PROCESS uses or receives the INPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
RECEIVED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

RECEIVES statement in PROCESS section.

Usage Rules:

- The names must be PROCESS names.
- An INPUT may be received by more than one PROCESS.

Synonyms:

RCVD DPNG DPG EC

Examples:

- RECEIVED BY P-104;
- RCVD P-89;
- RCVD P-90 DPG CONDITION-A EC ELEMENT-A;
- RCVE P-91 DEPENDING ON CONDITION-B;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- It may be used in any section except the PROBLEM-DEFINER section.

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
SECURITY IS security-name(s) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- It may be used in any section except a DEFINE section for a SOURCE.
- A name may have several SOURCES.

Synonyms:

SRC SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SUBPARTS Statement

Purpose:

To show the structural relationship of this INPUT to lower-lavel INPUT(S). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

```
SUBPARTS ARE input-name(s) ;
```

Complementary Statements:

PART statement in an INPUT section.

Usage Rules:

-The names must be INPUT names.

-An INPUT may be composed of several other INPUTS.

Synonyms:

SUBP

Examples:

```
- SUBPARTS ARE IN-101, IN-103;

- SUBP IN-309, INPUT-6785;
```

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to defined short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

```
SYNONYMS ARE synonym-name(s) ;
```

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE I-11, INPUT-11;
- SYNONYM IS INPUT-11;
- SYN ALPHA;

TERMINATES Statement

Purpose:

To specify a PROCESS/PROCESSES that are terminated by this INPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
     TERMINATES process-name(s)

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:

TERMINATED statement in PROCESS section.

Usage Rules:

- An INPUT may TERMINATE several PROCESSES. |S DPNG DPG ECYN| TRMS

Examples:

- TERMINATES PAYROLL-PROCESSING;
- TRMS PRINTING-PROCESS, PACKING-PROCESS
  DPNG ON ERROR-OCCURRENCE;
- TRMS PRINTING-PROCESS
  DPNG ALL-DONE FOR EACH EMPLOYEE-FILE;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

TRIGGERS Statement

Purpose:
To specify a PROCESS/PROCESSES that are triggered by this INPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
TRIGGERS process-name(s)

      [DEPENDING ON element-    name(s) ]
      [             condition-          ]

      [             group-              ]
      [             entity-             ]
      [FOR EACH     element-    name(s) ];
      [             output-             ]
      [             input-              ]
      [             set-                ]
```

Complementary Statements:
TRIGGERED statement in the PROCESS section.

Usage Rules:
- An INPUT may TRIGGER several PROCESSES.

Synonyms:

TRGS DPNG DPG EC

Examples:

```
- TRIGGERS MISSILE-CORRECTION, EVASIVE-MANEUVERS;

- TRGS MAIN-PROCESSING
           DEPENDING ON TIME-CARD-READY
           FOR EC EMPLOYEE-RECORD;

- TRGS PROCESS-A DPNG ELEMENT-A, ELEMENT-B;

- TRGS PROC-B EC INPUT-15;
```

USED Statement

Purpose:

To indicate the PROCESS(ES) that USE(D) this INPUT, and optionally, DERIVE(S) OUTPUTS or UPDATE(S) SETS, ENTITIES, GROUPS, or ELEMENTS, and to specify conditions and/or iterations associated with the DERIVE(s) or UPDATE(s) statements.

Syntax:

```
                             [                       set-                ]
                             [      { DERIVE }  *output-                 ]
USED BY process-name(s)      [  TO  {        }   entity-       name(s)   ]
                             [      { UPDATE }   group-                  ]
                             [                   element-                ]

        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:

USES, UPDATES or DERIVES statement in a PROCESS section and DERIVED or UPDATED statement in SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

-Several PROCESSES may use the INPUT.

-DEPENDING ON or FOR EACH statements can only be used with DERIVE or UPDATE clauses.

Synonyms:

DRV UPD DPNG DPG EC

Examples:

- USED BY PROCESS;

- USED BY LINEAR-PROCESS, INTEGER-PROCESS TO DERIVE ALPHA
          DPNG FUNCTION-TYPE;

- USED BY PROCESS-A DPNG CONDITION-B
          FOR EC ELEMENT-C;

INPUT SECTION

4.9 <u>INTERFACE Section Header Statement</u>

Purpose:

To allow a detailed description of an INTERFACE. The INTERFACE is an object, organization or system outside the boundaries of the target system that interacts with the system being described. It identifies the origin and destination of system products so that a complete understanding of the system may be obtained.

Syntax:

<u>INTERFACE</u> interface-name(s) ;

Usage Rules:

-Must be the first statement of every INTERFACE section.

-Several INTERFACES may be defined at once.

Synonyms:

INTF

INTERFACES

RWE

REAL-WORLD-ENTITY

ORGU

ORGANIZATIONAL-UNIT

Examples:

- INTERFACE RWE-22;
- RWE PAYROLL;
- ORGANIZATIONAL-UNIT STENO-POOL;
- ORGU WAREHOUSE-4;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value

             [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
 To specify properties or characteristics particular to a given section.

Syntax:

```
                               { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name       {           } [  ,attr-name     {           } ] ;
                               {  integer  } [                 {  integer  } ]
```

Complementary Statements:
 none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

GENERATES Statement

Purpose:
To give those INPUTS generated by this INTERFACE, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
GENERATES input-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:
GENERATED statement in INPUT section.

Usage Rules:
- The names must be INPUT names.
- A INTERFACE may generate several INPUTS.

Synonyms:

GENS DPNG DPG EC

Examples:
- GENERATES SYSTEM-IN-1;
- GENERATES IN-A, IN-B;
- GENS SYSTEM-INPUT DPENDING ON EMPLOYEE-STATUS;
- GENS SYS-A-IN,SYS-B-INDPG ELE=A, ELE-B
  FOR EACH SET-A, SET-B;

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

key	keyword

Examples:

- keyword is payroll;
- key is con-c1;
- keywords are emp, empl, employee;

PART Statement

Purpose:

To show the structural relationship of this INTERFACE to a higher-level INTERFACE. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

PART OF interface-name ;

Complementary Statements:

SUBPARTS statement in an INTERFACE section.

Usage Rules:

- The name must be an INTERFACE name.

- Only one INTERFACE name can be given, hence, only a tree structure may be established.

Synonyms:

none.

Examples:

- PART OF PAYROLL-SYSTEM;

- PART DEPT-601;

RECEIVES Statement

Purpose:

To identify the OUTPUTS produced by the system and show where they are used outside the system, and optionally, to specify conditions and/or iterations associated with RECEIVES statement. This is necessary for a complete system definition.

Syntax:

```
RECEIVES output-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:

RECEIVED BY statement in OUTPUT section.

Usage Rules:

-The names must be OUTPUT names.

-An INTERFACE may receive several OUTPUTS.

Synonyms:

RCVS DPNG DPG FC

Examples:

- RECEIVES FORECAST-FILE-OUTPUT;

- RECEIVES OUTPUT-FILE-A, OUTPUT-FILE-B DPNG ERROR-OCCURENCE;

- RCVS OUT-1001, OUT-103 DPG ERROR-OCCURENCE
  FOR EC FILE-A, FILE-B;

RESPONSIBLE Statement

Purpose:

To identify those SETS which this INTERFACE controls, maintains, and/or administers.

Syntax:

```
RESPONSIBLE FOR set-name(s) ;
```

Complementary Statements:

RESPONSIBLE-INTERFACE statement in SET section.

Usage Rules:

-The names must be SET names.

-An INTERFACE may be RESPONSIBLE for several SETS.

Synonyms:

RESP   RES

Examples:

```
- RESPONSIBLE FOR PAYROLL-FILE;

- RESP FILE-A, FILE-B;
```

<u>RESPONSIBLE-PROBLEM-DEFINER</u> <u>Statement</u>

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
SECURITY IS security-name(s) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SECURITY-ACCESS-RIGHT Statement

Purpose:
To give the type and level of security associated with an INTERFACE during operation of the target system.

Syntax:

```
SECURITY-ACCESS-RIGHT classification-name [ integer ]

                [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

SAR   SECURITY-ACCESS-RIGHTS

Examples:

- SECURITY-ACCESS-RIGHTS ARE PERSONNEL, SEC-LEVEL 3;
- SAR RING-LEVEL 2, UPDATE;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SUBPARTS Statement

Purpose:

To show the structural relationship of this INTERFACE to lower-level INTERFACE(S). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

SUBPARTS ARE interface-name(s) ;

Complementary Statements:

PART statement in an INTERFACE section.

Usage Rules:

- The names must be INTERFACE names.
- An INTERFACE may be composed of several other INTERFACES.

Synonyms:

SUBP

Examples:

- SUBPARTS ARE RWE-1, RWE-2;
- SUBP : PAYROLL-SYSTEM;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE I-11, INTERFACE-11;
- SYNONYM IS INTERFACE-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;

- TKEY part-1, part-2;

4.10 INTERVAL Section Header Statement

Purpose:

To allow a detailed description of an INTERVAL or INTERVALS. An INTERVAL is a specific duration of time or a time unit within the system. In defining frequency of an occurrence in the system, the frequency must be defined with respect to some time unit. For example, the designer might specify that a fiscal year lasted from June to May, and a calender year from January to December.

Syntax:

<u>INTERVAL</u> interval-name(s) ;

Usage Rules:

- It must be the first statement in an INTERVAL section.
- Several INTERVALS may be defined at once.

Synonyms:

INT INTERVALS

Examples:

- INTERVAL WORK-WEEK;
- INTERVALS: BUSINESS-DAY, DAY;
- INT PERIOD-1;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
   To specify properties or characteristics particular to a given section.

Syntax:

```
                              { attv-name } [                 { attv-name   ]
ATTRIBUTES ARE attr-name      {           } [ ,attr-name      {             ] ...
                              {  integer  } [                 {   integer } ]
```

Complementary Statements:
   none.

Usage Rules:
   -It may be used in any section.

   -A name may have several ATTRIBUTES

Synonyms:

   ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CONSISTS Statement

Purpose:

To describe the combination of other INTERVALS which make up this INTERVAL. This implies that each instance of the INTERVAL will contain values of other INTERVAL names. An INTERVAL may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
CONSISTS OF [ system-parameter ] interval-name

     [ , [ system-parameter ] interval-name ] ... ;
```

Complementary Statements:

None.

Usage Rules:

-The names, other than the SYSTEM-PARAMETERS , must be INTERVAL names.

-An INPUT may contain several INTERVALS.

Synonyms:

CSTS

Examples:

- CONSISTS OF INTERVAL-A;
- CONSISTS OF INTERVAL-1, INTERVAL-2;
- CSTS: SIXTY SECONDS, ONE HOUR;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

KEYWORDS ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY KEYWORD

Examples:

- KEYWORD IS PAYROLL;

- KEY IS CON-C1;

- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- It may be used in any section except the PROBLEM-DEFINER section.

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

<u>SEE-MEMO</u> <u>Statement</u>

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

<u>SEE-MEMO</u> memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it, is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a MEMO section, or a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE I-11, INTERVAL-11;

- SYNONYM IS INTERVAL-11;

- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.11 MEMO Section Header Statement

Purpose:

To define MEMOS. A MEMO is a description relevent to one or more other objects in the target system. MEMOS can be used to record as part of the system documentation significant information which needs to be highlighted. This might include assumptions made during design, limitations assumed or known to exist (e.g. Hardware. They can also be used to record outstanding problems, requests, effective dates, etc.

Syntax:

```
MEMO memo-name(s) ;
```

Usage Rules:

- It must be the first statement in a MEMO section.
- Several MEMOS may be defined at once.

Synonyms:

none.

Examples:

```
- MEMO NOTE-ON-UNRESOLVED-PROCESS-63;

- MEMO M-73, M-86;
```

APPLIES Statement

Purpose:
To tie this MEMO to one or more sections so that a cross-reference to the MEMO appears in the documentation.

Syntax:

APPLIES TO non-memo-name(s) ;

Complementary Statements:
SEE-MEMO statement in all sections except the MEMO section.

Usage Rules:
- The names may be any type of name except a MEMO name.

Synonyms:

APP

Examples:

- APPLIES TO PROCESS-1, PROCESS-2;
- APPLIES TO FREQUENCY-BAND, PRICING-UNIT-NAME;
- APP NETWORK-SOURCE;
- APP LINK-IDENT, NETWORK-NOTES, BASE-NETWORK;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                  [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;

ATTRIBUTES Statement

Purpose:

To specify properties or characteristics particular to a given section.

Syntax:

```
                               { attv-name } [                 { attv-name
ATTRIBUTES ARE attr-name {             } [ ,attr-name {
                               {  integer  } [                 {  integer
```

Complementary Statements:

none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
  THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;
DESC;
  ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY   KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- It may be used in any section except the PROBLEM-DEFINER section.

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC   SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-[illegible];

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE M-11, MEMO-11;

- SYNONYM IS MEMO-11;

- SYN ALPHA;

TRACE-KEY Statement

Purpose:
To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:
APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:
- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;

- TKEY part-1, part-2;

4.12 OUTPUT Section Header Statement

Purpose:

To allow a detailed description of an OUTPUT. An OUTPUT is used to describe a collection of information produced by the target system, but is used external to that system. The OUTPUT section is used to show the flow of data from the system to the outside world. Hence, it crosses the system boundary. It can also be used to locate and uniquely identify each system output.

Syntax:

```
OUTPUT output-name(s) ;
```

Usage Rules:

- Several OUTPUTS may be defined at a time.

Synonyms:

OUT

Examples:

- OUTPUT OUT-432;
- OUTPUT PAYROLL-CHECK;
- OUT OUT-431;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                                { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name {             } [ ,attr-name {             } ] .
                                {  integer  } [                 {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:
- It may be used in any section.
- A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CLASSIFICATION Statement

Purpose:
To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]
                [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

CLS   CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;

- CLS RING-LEVEL 2, UPDATE;

CONSISTS Statement

Purpose:

To describe the combination of GROUPS, and/or ELEMENTS which make up this OUTPUT. This implies that each instance of the OUTPUT will contain values of the GROUP and ELEMENT names. A GROUP or ELEMENT may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
                                               element-
CONSISTS OF [ system-parameter ]   group-name

                                                    element-
             [ , [ system-parameter ]   group-name ] ... ;
```

Complementary Statements:

CONTAINED statement in a GROUP or ELEMENT section.

Usage Rules:

-The names, other than the system-parameters, must be GROUP or ELEMENT names.

-An OUTPUT may contain several GROUPS or ELEMENTS.

Synonyms:

CSTS

Examples:

- CONSISTS OF TWO DATA-GROUP-1;
- CONSISTS: DATA-GROUP-1, ELEMENT-A;
- CSTS OF SPAN-ELEMENT-A;
- CSTS: GROUP-NO-1, GROUP-NO-2;

CONTAINED Statement

Purpose:

To give the SETS that contain this OUTPUT. An OUTPUT being contained in a SET means that the data values contained in the OUTPUT will be included in the logical SET.

Syntax:

CONTAINED IN set-name(s) ;

Complementary Statements:

CONSISTS statement in SET section.

Usage Rules:

- The names must be SET names.
- Several SETS may contain a given OUTPUT.

Synonyms:

CNTD

Examples:

- CONTAINED IN MASTER-FILE;
- CNTD: HS-1,HS-2;
- CNTD FILE-1;

DERIVED Statement

Purpose:
To give a PROCESS that DERIVES values for the OUTPUT and, optionally, the SETS, INPUTS, ENTITIES, GROUPS, and/or ELEMENTS used in the derivation, and to specify conditions and/or iterations associated with the action.

Syntax:

```
                                   [         group-              ]
                                   [        entity-              ]
DERIVED BY  process-name(s) [ USING       set-      name(s) ]
                                   [         input-              ]
                                   [       element-              ]

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:
DERIVES or USES statement in a PROCESS section and USED BY statement in a SET, INPUT, ENTITY, GROUP or ELEMENT section.

Usage Rules:
-Several PROCESSES may derive values for an OUTPUT.

Synonyms:

DRVD USG DPNG DPG EC

Examples:

- DERIVED BY PROCESS-A USING INPUT-1;

- DERIVED BY PROCESS-1 USING ENTITY-A, ENTITY-B;

```
- DRVD PROCESS-O USG INPUT-1 DPNG CONDITION-A;

- DRVD PROCESS-NAME USG ENTITY-A, GROUP-B
      DPNG ON CONDITION-A
      FOR EC SET-A, SET-B;
```

OUTPUT SECTION

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

GENERATED Statement

Purpose:

To identify the PROCESS which is responsible for producing this OUTPUT, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
GENERATED BY process-name(s)

        [DEPENDING ON element-     name(s) ]
        [             condition-           ]

        [             group-               ]
        [             entity-              ]
        [FOR EACH     element-     name(s) ];
        [             output-              ]
        [             input-               ]
        [             set-                 ]
```

Complementary Statements:

GENERATES statement in PROCESS section.

Usage Rules:

- The names must be PROCESS names.
- An OUTPUT can be GENERATED by more than one PROCESS.

Synonyms:

GEND DPNG DPG EC

Examples:

```
- GENERATED BY OUTPUT-PROCESS-1;

- GEND BY PROCESS-UPDATE DPNG CONDITION-A
           FOR EC ENTITY-A, ENTITY-B;
```

HAPPENS Statement

Purpose:

One purpose is to give the volume for this OUTPUT. More than one instance of an OUTPUT may occur over some period of time. The number of instances of the OUTPUT which occur in a time INTERVAL is expressed with this statement. Another purpose is to declare that an instance of the OUTPUT occurs repetitively with a specific cycle. Lastly, this statement may be used to specify that the OUTPUT occurs after some delay, or at a particular time.

Syntax:

```
                {system-parameter TIMES-PER interval-name}
        HAPPENS {EVERY system-parameter interval-name    };
                {[WITHIN] system-parameter interval-name }
                {                   AFTER event-name      }
```

Complementary Statements:

None.

Usage Rules:

-The statement may be given as many times as necessary for different INTERVALS.

-Combination of HAPPENS statements cannot be used for same INTERVAL name.

Synonyms:

HAP TIMP EVR EVY WI WTN WTH AF

Examples:

- HAPPENS TWELVE TIMES-PER INT-A;
- HAP THREE TIMP INT-2;
- HAP EVY ONE MONTH;

- HAP ONE DAY AF EVENT-A;

- HAP WTH TWO WEEKS AF EVENT-B;

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
        KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

```
- KEYWORD IS PAYROLL;

- KEY IS CON-C1;

- KEYWORDS ARE EMP, EMPL, EMPLOYEE;
```

PART Statement

Purpose:

To show the structural relationship of this OUTPUT to a higher-level OUTPUT. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

PART OF output-name ;

Complementary Statements:

SUBPARTS statement in an OUTPUT section.

Usage Rules:

-The name must be an OUTPUT name.

-Only one OUTPUT name can be given, hence, only a tree structure may be established.

Synonyms:

none.

Examples:

-PART OF OUTPUT-897;

RECEIVED Statement

Purpose:
To show which INTERFACE uses or receives the OUTPUT, and optionally, to specify conditions and/or iterations associated with the actions.

Syntax:

```
RECEIVED BY interface-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:
RECEIVES statement in INTERFACE section.

Usage Rules:
-The names must be INTERFACE names.

Synonyms:

RCVD DPNG DPG EC

Examples:

- RECEIVED BY PWF-104;

- RCVD DEPT-89 DPNG CONDITION-10;

- RCVE DEPT-100 DPNG ELEMENT-9
  FOR EC INPUT-A;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:
RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;

- SECURITIES ARE D-ORMISTON, S-MENNEL;

- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

SEE-MEMO memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;

- SOURCE: SDP-3-0;

AD-A060 780 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78 F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-130-VOL-2 NL

4 OF 6
AD
A060780

SUBPARTS Statement

Purpose:

To show the structural relationship of this OUTPUT to lower-level OUTPUT(S). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

```
SUBPARTS ARE output-name(s) ;
```

Complementary Statements:

PART statement in an OUTPUT section.

Usage Rules:

-The names must be OUTPUT names.

-An OUTPUT may be composed of several other OUTPUTS.

Synonyms:

SUBP

Examples:

```
- SUBPARTS ARE OUT-101, OUT-103;

- SUBP OUT-309, OUTPUT-897;
```

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

```
SYNONYMS ARE synonym-name(s) ;
```

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a MEMO section, or a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE O-11, OUTPUT-11;

- SYNONYM IS OUTPUT-11;

- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.13 PROBLEM-DEFINER Section Header Statement

Purpose:

To define a PROBLEM-DEFINER or DEFINERS. The PROBLEM-DEFINER is the person responsible for one or more URL object definitions. This section identifies for which other sections within the documentation the PROBLEM-DEFINER has responsibility. This is useful in establishing good documentation controls for the system.

Syntax:

PROBLEM-DEFINER problem-definer-name(s) ;

Usage Rules:

-Must be the first statement in a PROBLEM DEFINER section.

-Several PROBLEM-DEFINERS may be defined at once.

Synonyms:

PD	PROBLEM-DEFINERS

Examples:

- PROBLEM-DEFINER J-SURTES;
- PROBLEM-DEFINERS: P-REZK, J-SMITH;
- PD: E-WINTERS;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value

                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
    To specify properties or characteristics particular to a given section.

Syntax:

```
                                 { attv-name } [                  { attv-name }  ]
ATTRIBUTES ARE attr-name {             } [ ,attr-name {             }  ]...
                                 {  integer  } [                  {  integer  }  ]
```

Complementary Statements:
    none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:
    To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:
    None.

Usage Rules:
    - See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

    DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
    KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY   KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MAILBOX Statement

Purpose:

To identify the location or address where this PROBLEM-DEFINER may be reached.

Syntax:

MAILBOX IS mailbox-name ;

Complementary Statements:

APPLIES statement in DEFINE section for a MAILBOX.

Usage Rules:

- The name must be a MAILBOX name.
- A PROBLEM-DEFINER may only have one MAILBOX.

Synonyms:

BOX	MBX	MAILBOXES

Examples:

- MAILBOX IS USERID-AA110;
- BOX IS FOUR-FORTY-FIVE-HAMILTON-AVE;
- MBX IS FIVE-WORLD-TRADE-CENTER;

__RESPONSIBLE__ __Statement__

Purpose:

To give the sections for which a PROBLEM-DEFINER is responsible.

Syntax:

__RESPONSIBLE__ FOR name(s) ;

Complementary Statements:

RESPONSIBLE-PROBLEM-DEFINER statement.

Usage Rules:

-The names may be any type of name except a PROBLEM-DEFINER name or a MAILBOX name.

-Only one PROBLEM-DEFINER may be RESPONSIBLE for any section.

Synonyms:

RESP	RES

Examples:

- RESPONSIBLE FOR P-101;

- RESP FOR P-10,P-11,P-12,P-13,P-14;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
SECURITY IS security-name(s) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORNISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

<u>SOURCE Statement</u>

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

<u>SOURCE</u> IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a MEMO section, or a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN SYNONYM

Examples:

- SYNONYMS ARE P-11, PROBLEM-DEFINER-11;

- SYNONYM IS PROBLEM-DEFINER-11;

- SYN ALPHA;

TRACE-KEY Statements

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.14 PROCESS Section Header Statement

Purpose:

To allow a detailed description of a PROCESS or PROCESSES. This section is used to show how data is used within the target system. For instance, a PROCESS can validate INPUTS, produce OUTPUTS , store and manipulate data to meet the objectives of the system, and cause the initiation of additional PROCESS(ES). It is also used to show the structure of the system and its component subsystems.

Syntax:

PROCESS process-name(s) ;

Usage Rules:

-Must be the first statement in a PROCESS section.

-Several PROCESSES may be defined at once.

Synonyms:

PROC PRC

Examples:

- PROCESS P-101;

- PROC P-32, P-86;

- PROCESS P-789,P-539;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value

                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:

Syntax:

```
                                    { attv-name } [                  { attv-name } ]
     ATTRIBUTES ARE attr-name       {           } [  ,attr-name      {           } ]...
                                    {  integer  } [                  {  integer  } ]
```

Complementary Statements:

none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DERIVES Statement

Purpose:
To give the data which is DERIVED by this PROCESS , and, optionally, the data used to DERIVE it, and conditions and/or iteration association with the derivation.

Syntax:

```
                   set-           [              set-           ]
                output-           [            input-           ]
    DERIVES   element-name(s)     [  USING   element-name(s)    ]
                 entity-          [           entity-           ]
                  group-          [            group-           ]

         [DEPENDING ON element-    name(s) ]
         [             condition-          ]

         [             group-              ]
         [             entity-             ]
         [FOR EACH     element-    name(s) ];
         [             output-             ]
         [             input-              ]
         [             set-                ]
```

Complementary Statements:
DERIVED or USED BY statements in SET, ELEMENT, ENTITY, GROUP, or OUTPUT sections and USES statement in PROCESS section.

Usage Rules:

-A single PROCESS may DERIVE several different SETS, OUTPUTS, ELEMENTS, ENTITIES, or GROUPS.

Synonyms:

DRVS USG DPNG DPG EC

Examples:

```
- DERIVES ELEMENT-407-X USING ELEMENT-407-Y;

- DERIVES ELEMENT-147 USING ELEMENT-48, ELEMENT-49, ELEMENT-50;

- DRVS ELE-22 USG ELE-221 DPNG CONDITION-A;

- DRVS ELE-186 USG ELE-1, ELE-17, ELE-23
       DPNG COND-A, COND-B
       FOR EC INPUT-105;
```

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

GENERATES Statement

Purpose:

To give those OUTPUTS which are GENERATED by this PROCESS, and optionally, to sepcify conditions and/or iterations associated with the action.

Syntax:

```
GENERATES output-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

GENERATED statement in OUTPUT section.

Usage Rules:

-The names must be OUTPUT names.

Synonyms:

GENS DPNG DPG EC

Examples:

```
- GENERATES FIRST-OUTPUT;

- GENERATES OUTPUT-1, OUTPUT-2;

- GENS OUT-A DPNG COND-A;

- GENS OUT-A,OUT-B DPNG COND-B
          FOR EC INPUT-1, INPUT-2;
```

HAPPENS Statement

Purpose:

One purpose is to give the number of times the PROCESS is used per INTERVAL. More than one instance of a PROCESS may occur over some period of time. The number of instances of the PROCESS which occur in a time INTERVAL is expressed with this statement. Another purpose is to declare that a PROCESS is used repetitively in a specific cycle. Lastly, this statement may be used to specify a delay or a particular time that the PROCESS may occur.

Syntax:

```
        {system-parameter TIMES-PER interval-name}
HAPPENS {EVERY system-parameter interval-name    };
        {[WITHIN] system-parameter interval-name }
        {                 AFTER event-name       }
```

Complementary Statements:

None.

Usage Rules:

-The statement may be given as many times as necessary for different INTERVALS.

-Combination of HAPPENS statements cannot be used for same INTERVAL name.

Synonyms:

HAP TIMP EVR EVY WI WTN WTH AF

Examples:

- HAPPENS SIX TIMES-PER NEW-INTERVAL;
- HAP ONE TIMP OLD-DATE-INT;
- HAP EVY ONE MONTH;
- HAP ONE DAY AF EVENT-A;
- HAP WTN TWO WEEKS AF EVENT-B;

INCEPTION-CAUSES Statement

Purpose:
To link an EVENT or EVENTS to the inception of the PROCESS, and optionally, to specify conditions and/or iteration associated with the action.

Syntax:

```
INCEPTION-CAUSES event-name(s)

        [DEPENDING ON element-    name(s) ]
        [             condition-          ]

        [             group-              ]
        [             entity-             ]
        [FOR EACH     element-    name(s) ];
        [             output-             ]
        [             input-              ]
        [             set-                ]
```

Complementary Statements:
INCEPTION statement in an EVENT section.

Usage Rules:
- The names must be EVENT names.
- A PROCESS may initiate several EVENTS.

Synonyms:
INCC DPNG DPG EC

Examples:
- INCEPTION-CAUSES UPDATE-EVT;
- INCC EVENT-1,EVENT-2 DPNG ON CONDITION-A;
- INCC EVENT-3 DPNG COND-B
FOR EC ENT-105;

INTERRUPTED Statement

Purpose:

To specify an EVENT/EVENTS, INPUT/INPUTS, or PROCESS/PROCESSES which interrupt this PROCESS, and optionally, to specify conditions and/or iterations associated with the interruption. Also, to specify CONDITIONS for which changes of state will cause interruption of this PROCESS.

Syntax:

```
                     event-
INTERRUPTED BY       input-name(s)
                   process-

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]


                                               { TRUE  }
INTERRUPTED WHEN condition-name BECOMES        {       };
                                               { FALSE }
```

Complementary Statements:

INTERRUPTS statement in the EVENT, INPUT, and PROCESS sections, and BECOMING INTERRUPTS statement in the CONDITION section.

Usage Rules:

- A PROCESS may be INTERRUPTED by several EVENTS, INPUTS, or PROCESSES.

- Only one CONDITION may be specified in a single statement. Separate statements are required for each CONDITION.

Synonyms:

```
     INTD DPNG DPG EC
```

Examples:

```
     - INTERRUPTED BY PURCHASE-ORDER-DELAY;

     - INTD HIGH-PRIO-INPUT, NEW-TASK-INPUT
              DPGN ON CONDITION-A FOR EC SET-10;

     - INTERRUPTED WHEN END-OF-FILE BECOMES FALSE ;

     - INTD WHEN MACHINE-BREAKDOWN T;
```

INTERRUPTS Statement

Purpose:

To specify PROCESS(ES) which are interrupted by this PROCESS, and optionally, to specify conditions and/or iterations associated with the interruptions.

Syntax:

```
INTERRUPTS process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

INTERRUPTED statement in the PROCESS section.

Usage Rules:

- A PROCESS may INTERRUPT several other PROCESSES.

Synonyms:

INTS DPNG DPG EC

Examples:

```
- INTERRUPTS SUBPROCESS-A, SUBPROCESS-B;
INTS SWITCHING-OPERATION DPNG ON HARDWARE-COND;

- INTS PROCESS-A DPG FIE-A, FIE-B
        FOR EC INPUT-10;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

```
-KEY ON-LINE PROCESS;

-KEYWORD TERMINAL;
```

MAINTAINS Statement

Purpose:

To give the RELATIONS and SUBSETTING-CRITERIA which are MAINTAINED by this PROCESS, and optionally, to specify conditions and/or iteration associated with the action.

Syntax:

```
                          relation-
   MAINTAINS subsetting-criteria-name(s)

         [DEPENDING ON element-    name(s) ]
         [             condition-          ]

         [             group-              ]
         [             entity-             ]
         [FOR EACH     element-    name(s) ];
         [             output-             ]
         [             input-              ]
         [             set-                ]
```

Complementary Statements:

MAINTAINED statement in DEFINE section for SUBSETTING-CRITERION, and MAINTAINED statement in RELATION section.

Usage Rules:

- The names must be either RELATION or SUBSETTING-CRITERIA names.
- A PROCESS may MAINTAIN several RELATIONS and SUBSETTING-CRITERIA.

Synonyms:

MTNS DPNG DPG EC

Examples:

```
- MAINTAINS RELATION-SET;

- MTNS FIRST-RELATION, FIFTY-FIRST-SET
        DPNG ON ELE-A, ELE-B
        FOR EC INPUT-100;
```

MAKES Statement

Purpose:
To give CONDITION(S) whose states are set by this PROCESS, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
                                 { TRUE  }
     MAKES condition-name(s)     {       }
                                 { FALSE }

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:
MADE statement in the CONDITION section.

Usage Rules:
- A PROCESS may MAKE several CONDITIONS become either TRUE or FALSE.

- A PROCESS cannot MAKE some CONDITIONS TRUE and some CONDITIONS FALSE in a single statement. Separate statements are required.

Synonyms:

MAK DPNG DPG EC

Examples:

```
- MAKES PROCESS-COMPLETION TRUE;
- MAK INPUT-READ, PRODUCTION-BEGAN F
          DPG ON COND-A
```

FOR FC SET-10;

PART Statement

Purpose:

To show the structural relationship of this PROCESS to a higher-level PROCESS. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

PART OF process-name ;

Complementary Statements:

SUBPARTS statement in a PROCESS section.

Usage Rules:

-The name must be a PROCESS name.

-Only one PROCESS name may be given, hence, only a tree structure can be established.

Synonyms:

none.

Examples:

-PART OF PAYROLL-SYSTEM;

PERFORMED Statement

Purpose:

To give the PROCESSOR that performs the PROCESS.

Syntax:

PERFORMED BY processor-name ;

Complementary Statements:

PERFORMS statement in PROCESSOR section.

Usage Rules:

- Only one PROCESSOR name may be given.

Synonyms:

PFMD

Examples:

- PERFORMED BY CPU-1;
- PFMD PROCESSOR-NO-1;

PROCEDURE Statement

Purpose:
To describe the sequence of operations needed to implement this PROCESS.

Syntax:

```
PROCEDURE :
        comment-entry ;
```

Complementary Statements:
None.

Usage Rules:
-Only one PROCEDURE statement may be given for any PROCESS.

Synonyms:

PRCD PRD

Examples:

```
- PROCEDURE;

    1. READ THE DATA FROM THE FILE

    2. CHECK TRANSACTION CODE

    3. CALL APPROPRIATE TRANSACTION PROCESS;

- PRCD;

    ANY RELEVANT COMMENTS TO AID THE PROGRAM DESIGNER;
```

RECEIVES Statement

Purpose:
To give the INPUTS RECEIVED by this PROCESS, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
RECEIVES input-name(s)

        [DEPENDING ON element-     name(s) ]
        [              condition-          ]

        [              group-              ]
        [              entity-             ]
        [FOR EACH      element-    name(s) ];
        [              output-             ]
        [              input-              ]
        [              set-                ]
```

Complementary Statements:
RECEIVED statement in INPUT section.

Usage Rules:
-The names must be INPUT names.

-A PROCESS may RECEIVE more than one INPUT.

Synonyms:

RCVS DPNG DPG FC

Examples:

```
- RECEIVES INPUT-100;

- RECEIVES INPUT-4A, INPUT-4B DPNG ELEMENT-A;

- RCVS INPUT-A100 DPNG COND-A, COND-B
        FOR EC SET-10, SET-20;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

RESOURCE-USAGE Statement

Purpose:
To give a pair of resource-usage parameter and resource usage parameter value for the PROCESS.

Syntax:

```
     RESOURCE-USAGE :
               system-parameter FOR resource-usage-parameter-name;
```

Complementary Statements:
RESOURCE-USAGE-PARAMETER-VALUE statement in RESOURCE-USAGE-PARAMETER section.

Usage Rules:
- The second term (system-parameter or number) is called the "resource-usage-parameter-value" (rup-value) for the resource-usage-parameter. A PROCESS may have several pairs of resource-usage-parameter-values as long as the resource usage parameters are not the same.

Synonyms:

RU

Examples:

- RESOURCE-USAGE: 10 FOR COMPLEXITY-RATING;

- RU 2000 FOR STATEMENTS-IN-PL;

- RU MAXIMUM-RATING RATING;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

<u>SECURITY</u> IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SECURITY-ACCESS-RIGHT Statement

Purpose:
To give the type and level of security associated with a PROCESS during operation of the target system.

Syntax:

```
SECURITY-ACCESS-RIGHT classification-name [ integer ]

                [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

SAF	SECURITY-ACCESS-RIGHTS

Examples:

- SECURITY-ACCESS-RIGHTS ARE PERSONNEL, SEC-LEVEL 3;
- SAF RING-LEVEL 2, UPDATE;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;

- SOURCE: SDP-3-0;

SUBPARTS Statement

Purpose:

To show the structural relationship of this PROCESS to lower-level PROCESS(ES). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

SUBPARTS ARE process-name(s) ;

Complementary Statements:

PART statement in a PROCESS section.

Usage Rules:

- The names must be PROCESS names.
- A PROCESS may be composed of several other PROCESSES.

Synonyms:

SUBP

Examples:

- SUBPARTS ARE P-101, P-103;
- SUBP P-309, INPUT-EDIT-PROCESS;

SYNONYMS Statement

Purpose:
To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

```
SYNONYMS ARE synonym-name(s) ;
```

Complementary Statements:
DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE P-11, PROCESS-11;
- SYNONYM IS PROCESS-11;
- SYN ALPHA;

TERMINATED Statement

Purpose:

To specify EVENT(S), INPUT(S), and/or PROCESS(ES) which terminate this PROCESS, and optionally, to specify conditions and/or iterations associated with the termination. Also, to specify CONDITIONS for which changes of state will terminate this PROCESS.

Syntax:

```
                              event-
     TERMINATED BY           input-name(s)
                            process-

          [DEPENDING ON element-     name(s) ]
          [              condition-          ]

          [              group-              ]
          [              entity-             ]
          [FOR EACH      element-    name(s) ];
          [              output-             ]
          [              input-              ]
          [              set-                ]


                                                   { TRUE  }
     TERMINATED WHEN condition-name BECOMES        {       } ;
                                                   { FALSE }
```

Complementary Statements:

TERMINATES statement in the EVENT, INPUT, and PROCESS sections, and BECOMING TERMINATES statement in the CONDITION section.

Usage Rules:

- A PROCESS may be TERMINATED by several EVENTS, INPUTS, or PROCESSES.

- Only one CONDITION may be specified in a single statement. Separate statements are required for each CONDITION.

Synonyms:

TRMD DPNG DPG EC

Examples:

- TERMINATED BY END-OF-INPUT;

- TRMD BY LAST-INPUT, NEW-ORDER-INPUT DPNG COND-A;

- TRMD ERROR-PROC, SEARCH-PROC DPNG COND-B EC INPUT-10;

- TRMD WHEN FATAL-ERROR BECOMES FALSE;

TERMINATES Statement

Purpose:
To specify a PROCESS/PROCESSES that are terminated by this PROCESS, and optionally, to specify conditions and/or iterations associated with the termination.

Syntax:

```
TERMINATES process-name(s)

      [DEPENDING ON element-   name(s) ]
      [             condition-         ]

      [             group-             ]
      [             entity-            ]
      [FOR EACH     element-   name(s) ];
      [             output-            ]
      [             input-             ]
      [             set-               ]
```

Complementary Statements:
TERMINATED statement in PROCESS section.

Usage Rules:
- A PROCESS may TERMINATE several other PROCESSES.

Synonyms:

TRMS DPNG DPG EC

Examples:

```
- TERMINATES OUTPUT-PRODUCTION;

- TRMS SET-UP-PROC, ERROR-CHECKING
          DPNG ON COND-A
          FOR EC INPUT-100, INPUT-200;
```

TERMINATION-CAUSES Statement

Purpose:

To indicate which EVENT or EVENTS occur when this PROCESS finishes, and to specify conditions and/or iterations associated with the action.

Syntax:

```
TERMINATION-CAUSES event-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:

TERMINATION statement in an EVENT section.

Usage Rules:

- The names must be EVENT names.
- A PROCESS may terminate several different EVENTS.

Synonyms:

TERC DPNG DPG EC

Examples:

```
- TERMINATION-CAUSES UPDATE-EVENT;

- TERC ISSUE-CHECK-EVENT DPNG ELE-A, ELE-B
       FOR EC EMPLOYEE-FILE;
```

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

<u>TRACE-KEY</u> trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

TRIGGERED Statement

Purpose:
To give the EVENT/EVENTS, INPUT/INPUTS, and PROCESS/PROCESSES which can TRIGGER this PROCESS, and optionally, to specify conditions and/or iterations associated with the action. Also, to specify a CONDITION which may trigger this PROCESS.

Syntax:

```
                      event-
    TRIGGERED BY     input-name(s)
                    process-

         [DEPENDING ON element-     name(s) ]
         [             condition-           ]

         [             group-               ]
         [             entity-              ]
         [FOR EACH     element-     name(s) ];
         [             output-              ]
         [             input-               ]
         [             set-                 ]


                                              { TRUE  }
    TRIGGERED WHEN condition-name BECOMES     {       } ;
                                              { FALSE }
```

Complementary Statements:
TRIGGERS statement in EVENT, INPUT, and PROCESS sections, and BECOMING TRIGGERS statement in the CONDITION section.

Usage Rules:
- Only one CONDITION may be specified in a single statement. A separate statement is necessary for each CONDITION specified.

-Several triggering EVENTS, INPUTS, or PROCESSES may be given.

Synonyms:

TRGD DPNG DPG EC

Examples:

```
- TRIGGERED BY UPDATE-EVENT;

- TRGD ORDER-PROC, ERROR-CHECKING, INFO-RETRIEVAL-PROC
        DPNG ON FLE-A FOR EC ORDER;

- TRIGGERED WHEN DATA-FOUND BECOMES TRUE;
```

TRIGGERS Statement

Purpose:
To specify a PROCESS/PROCESSES which are triggered by this PROCESS, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
     TRIGGERS process-name(s)

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:
TRIGGERED statement in the PROCESS section.

Usage Rules:
- A PROCESS may TRIGGER several other PROCESSES.

Synonyms:

TRGS DPNG DPG EC

Examples:

```
- TRIGGERS MAIN-PROCESSING;

- TRGS INPUT-CHECKING, MAIN-PROCESSING
       DPNG ON ELE-A, ELE-B
       FOR EC INPUT;
```

UPDATES Statement

Purpose:

To give the ENTITIES, GROUPS, ELEMENTS and/or SETS which are updated by this PROCESS, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
                  group-          [          group-           ]
                  entity-         [          entity-          ]
       UPDATES    element- name(s) [ USING   element-  name(s) ]
                  set-            [          set-             ]
                                  [          input-           ]

         [DEPENDING ON element-   name(s) ]
         [             condition-         ]

         [             group-             ]
         [             entity-            ]
         [FOR EACH     element-   name(s) ];
         [             output-            ]
         [             input-             ]
         [             set-               ]
```

Complementary Statements:

UPDATED or USED BY statements in ENTITY, GROUP, ELEMENT and SET sections and USES statement in PROCESS section.

Usage Rules:

none.

Synonyms:

UPDS USG DPNG DPG EC

Examples:

```
- UPDATES HS-SEGMENT, HT-SEGMENT;

- UPDS AQ-SEGMENT USING E-2, E-5
        DPNG ON COND-A
        FOR EC ELEMENT-101;
```

USES Statement

Purpose:
To give those SETS, GROUPS, ELEMENTS, INPUTS and ENTITIES used by the PROCESS, and optionally, to specify conditions and/or iterations associated with DERIVE or UPDATE statement.

Syntax:

```
             set-            [                      set-            ]
           input-            [     { DERIVE }   *output-            ]
USES     element-name(s)     [ TO  {        }    element-   name(s) ]
           group-            [     { UPDATE }      group-           ]
          entity-            [                    entity-           ]

   [DEPENDING ON element-   name(s) ]
   [             condition-         ]

   [             group-             ]
   [             entity-            ]
   [FOR EACH     element-   name(s) ];
   [             output-            ]
   [             input-             ]
   [             set-               ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:
USED, UPDATED or DERIVED statement in a SET, GROUP, ELEMENT or ENTITY section and DERIVES or UPDATES statement in PROCESS section.

Usage Rules:
- A PROCESS may use several different SETS, GROUPS, ELEMENTS, INPUTS or ENTITIES.

- DEPENDING ON or FOR EACH statements can only be used with DERIVE or UPDATE clauses.

Synonyms:

DRV UPD DPNG DPG EC

Examples:

```
     - USES TASK-FILE;

     - USES PERSONNEL-FILE, PAYROLL-FILE
                TO DERIVE PAYCHECK-OUTPUT
                FOR EC INPUT-TIME-CARD;
```

UTILIZED Statement

Purpose:
To show the structural relationship of this PROCESS to higher-level PROCESSES. and optionally, to specify conditions and/or iterations associated with the utiliziation. This staement allows PROCESSES to be used by more than one higher-level PROCESS.

Syntax:

```
UTILIZED BY process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:
UTILIZES statement in the PROCESS section.

Usage Rules:
- The names must be PROCESS names.

- A PROCESS may be UTILIZED by several PROCESSES

Synonyms:

UTLD DPNG DPG EC

Examples:

- UTILIZED LP-ALGORITHM;

- UTILIZED COMMON-INPUT-PROCESS, COMMON-OUTPUT-PROCESS;

- UTLD: TAPE-READ-PROCESS DPNG COND-TAPE-READY;

```
    - UTID: UPDATE-BILL-PROC-1,UPDATE-BILL-PROC-2
            DPNG ON FLE-A, FLE-B
            FOR EC SET-10;
```

UTILIZES Statement

Purpose:
To show the structural relationship of this PROCESS to lower-level PROCESSES, and optionally, to specify conditions and/or iterations associated with the utilization. This statement allows several higher-level PROCESSES to share the use of the same lower-level PROCESS.

Syntax:

```
UTILIZES process-name(s)

     [DEPENDING ON element-    name(s) ]
     [             condition-          ]

     [             group-              ]
     [             entity-             ]
     [FOR EACH     element-    name(s) ];
     [             output-             ]
     [             input-              ]
     [             set-                ]
```

Complementary Statements:
UTILIZED statement in the PROCESS section.

Usage Rules:
-The names must be PROCESS names.

-A PROCESS may UTILIZE several PROCESSES

Synonyms:

UTLS DPNG DPG EC

Examples:

```
    - UTILIZES LP-ALGORITHM;

    - UTILIZES COMMON-INPUT-PROCESS, COMMON-OUTPUT-PROCESS;

    - UTLS: TAPE-READ-PROCESS DPNG INPUT-TYPE;

    - UTLS: UPDATE-BILL-PROC-1,UPDATE-BILL-PROC-2
            DPNG ON ELE-A, ELE-B
            FOR EC SET-10;
```

4.15 PROCESSOR Section Header Statement

Purpose:

To allow a detailed description of a PROCESSOR.

Syntax:

PROCESSOR processor-name(s);

Usage Rules:

- Must be the first statement in a PROCESSOR section.
- More than one PROCESSOR may be defined at once.

Synonyms:

PROCR PRCR PROCESSORS

Examples:

- PROCESSOR PP-1;
- PRCR CPU, DISK-MEMORY;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                         { attv-name } [            { attv-name } ]
ATTRIBUTES ARE attr-name {           } [ ,attr-name {           } ] .
                         {  integer  } [            {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;

- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;

- ATTR CHAR ZZZ9V9;

<u>CONSUMES</u> <u>Statement</u>

Purpose:

To give the resource consumption value for the PROCESSOR.

Syntax:

<u>CONSUMES</u> resource-name AT <u>RATE</u> OF

system-parameter <u>PER</u> resource-usage-parameter-name;

Complementary Statements:

CONSUMED statement in RESOURCE section.

Usage Rules:

- A name may have several CONSUMES statements as long as they are not contradictory, i.e. , at most one CONSUMED statement is allowed for a unique pair of resource-name and resource-usage-parameter-name.

Synonyms:

CNSS

Examples:

- CONSUMES REAL TIME AT A RATE OF 10 PER NUMBER-OF-CHARACTERS;

- CNSS DOLLARS RATE X PER DIFFICULTY-GRADING;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION :
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
     KEYWORDS ARE keyword-name(S) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

```
- KEYWORD IS PAYROLL;

- KEY IS CON-C1;

- KEYWORDS ARE EMP, EMPL, EMPLOYEE;
```

PART Statement

Purpose:

To show the structural relationship of this PROCESSOR to a higher level PROCESSOR. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

PART OF processor-name:

Complementary Statements:

SUBPARTS statement in PROCESSOR section.

Usage Rules:

- Only one PROCESSOR name may be given, hence only a tree structure can be established.

Synonyms:

None.

Examples:

- PART OF MACHINES;

PERFORMS Statement

Purpose:
To give the PROCESSES that the PROCESSOR performs.

Syntax:

PERFORMS process-name(s);

Complementary Statements:
PERFORMED statement in PROCESS section.

Usage Rules:
- More than one PROCESS may be performed by a PROCESSOR, but a PROCESS may be performed by one PROCESSOR only.

Synonyms:

PFMS

Examples:

- PERFORMS PAYROLL-PROCESSING;

- PFMS PROCESS-A, PROCESS-B;

<u>RESPONSIBLE-PROBLEM-DEFINER Statement</u>

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

<u>RESPONSIBLE-PROBLEM-DEFINER</u> IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SECURITY-ACCESS-RIGHT Statement

Purpose:
To give the type and level of security associated with a PROCESSOR during operation of the target system.

Syntax:

```
SECURITY-ACCESS-RIGHT classification-name [ integer ]
                      [, classification-name [ integer ]]... ;
```

Complementary Statements:
None.

Usage Rules:
- The name must be a CLASSIFICATION name.

Synonyms:

SAR	SECURITY-ACCESS-RIGHTS

Examples:

- SECURITY-ACCESS-RIGHTS ARE PERSONNEL, SEC-LEVEL 3;
- SAR RING-LEVEL 2, UPDATE;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

<u>SEE-MEMO</u> memo-name(s) ;

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SPP-3-0;

SUBPARTS Statement

Purpose:

To show the structural relationship of this PROCESSOR to lower-level PROCESSORS. This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

SUBPARTS ARE processor-name(s);

Complementary Statements:

PART statement in PROCESSOR section.

Usage Rules:

- A PROCESSOR may be composed of several other PROCESSORS.

Synonyms:

SUBP

Examples:

- SUBPARTS ARE HUMAN,MACHINES;
- SUBP PR-1, PR-2, PR-3;

SYNONYMS Statement

Purpose:
To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

```
SYNONYMS ARE synonym-name(s) ;
```

Complementary Statements:
DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE P-11, PROCESSOR-11;
- SYNONYM IS PROCESSOR-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.16 RELATION Section Header Statement

Purpose:

To define a RELATION or RELATIONS. This section shows how two ENTITIES are logically connected. Examples of relations are husband-to-wife or employee-to-company.

Syntax:

RELATION relation-name(s) ;

Usage Rules:

-Must be the first statement of every RELATION section.

-Several RELATIONS may be defined at once.

Synonyms:

RLN RELATIONS

Examples

- RELATION NH-RELATION;

- RLN NI-RELATION, NS-RELATION;

- RELATIONS REL-1, REL-2, REL-3;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value

                   [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ASSOCIATED-DATA Statement

Purpose:

To give those GROUPS and/or ELEMENTS which are the result of the RELATION being described or which describe the RELATION. Although the data may be contained in either or both ENTITIES. ASSOCIATED-DATA does not belong to either ENTITY RELATION being described. ASSOCIATED DATA does not belong to either ENTITY exclusively, but to both jointly.

Syntax:

```
                         group-
ASSOCIATED-DATA IS element-name(s) ;
```

Complementary Statements:

ASSOCIATED statement in ELEMENT and GROUP section.

Usage Rules:

- The names must be either ELEMENT or GROUP names.
- The ELEMENTS associated with a RELATION may not be part of an ENTITY.

Synonyms:

ASCD

Examples:

- ASSOCIATED-DATA IS SPAN-SEGMENT;
- ASSOCIATED-DATA IS ELE-1,ELE-2,GROUP-9;
- ASCD LINK-SEGMENT;
- ASCD ELEMENT-A,GROUP-9;

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                               { attv-name } [                  { attv-name }  ]
ATTRIBUTES ARE attr-name {             } [ ,attr-name {             }  ] ..
                               {  integer  } [                  {  integer  }  ]
```

Complementary Statements:
none.

Usage Rules:
-It may be used in any section.

-A name may have several ATTRIBUTES

Synonyms:

ATTR   ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;

- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;

- ATTR CHAR ZZZ9V9;

BETWEEN Statement

Purpose:
To give the ENTITIES which are related, e.g. Logically connected, via a particular RELATION.

Syntax:

BETWEEN entity-name AND entity-name ;

Complementary Statements:
RELATED statement in ENTITY section.

Usage Rules:
- Both names must be ENTITY names, they may, however, be the same ENTITY name.
- All RELATIONS are binary.
- All RELATIONS must have exactly one BETWEEN statement which gives the ENTITIES involved in the RELATION.

Synonyms:

BTWN

Examples:

- BETWEEN WOMAN AND MAN;
- BETWEEN ENTITY-1 AND ENTITY-2 ;
- BETWEEN RECORD-1 AND RECORD-2;
- BTWN EMP-INFO JOB-INFO ;

CARDINALITY Statement

Purpose:
To define the number of times this RELATION applies in the system.

Syntax:

```
CARDINALITY IS system-parameter ;
```

Complementary Statements:
None.

Usage Rules:
-A RELATION may have only one CARDINALITY.

Synonyms:

CARD OCCS OCCURRENCES

Examples:

- CARDINALITY IS TWENTY;

- CARD FORTY-SEVEN;

CONNECTIVITY Statement

Purpose:
To define the number of occurrences in the RELATION of one ENTITY with respect to the other. For example, one could specify that there is one company-entity related to many employee-entities.

Syntax:

CONNECTIVITY IS system-parameter TO system-parameter ;

Complementary Statements:
None.

Usage Rules:
- Any RELATION may have only one CONNECTIVITY given.

Synonyms:

CONN

Examples:

- CONNECTIVITY IS ONE TO ONE;

- CONN MANY TO TWO;

DERIVATION Statement

Purpose:
To give the DERIVATION rules for those RELATIONS which are derivable for the data. This implies that the RELATION being described is a DERIVED RELATION, not a direct RELATION.

Syntax:

```
DERIVATION :
      comment-entry ;
```

Complementary Statements:
None.

Usage Rules:
- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DRVN

Examples:

```
DERIVATION;
   THIS RELATIONSHIP EXISTS TO SHOW HOW UPON ENTRY OF THE TIME
CARD AN UPDATE OCCURS;

DRVN;
   ANY RELEVANT COMMENTS MAY BE ENTERED;
```

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
    THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
    ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY   KEYWORD

Examples:

- KEYWORD IS PAYROLL;

- KEY IS CON-C1;

- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MAINTAINED Statement

Purpose:

To designate those PROCESSES which change the instances of the ENTITIES that are connected by the RELATION, and optionally, to specify conditions and/or iterations associated with the action.

Syntax:

```
MAINTAINED BY process-name(s)

    [DEPENDING ON element-    name(s) ]
    [             condition-          ]

    [             group-              ]
    [             entity-             ]
    [FOR EACH     element-    name(s) ];
    [             output-             ]
    [             input-              ]
    [             set-                ]
```

Complementary Statements:

MAINTAINS statement in PROCESS section.

Usage Rules:

- The names must be process- names.
- A RELATION may be MAINTAINED BY more than one PROCESS.

Synonyms:

MIND DPNG DPG EC

Examples:

```
- MAINTAINED BY PROCESS-6543;

- MTND P-18,P-190 DPNG COND-1
        FOR EC FILE-A, FILE-B;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

<u>SECURITY</u> IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:
To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:
APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO RW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:
To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:
APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- It may be used in any section except a DEFINE section for a SOURCE.

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;

- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN SYNONYM

Examples:

- SYNONYMS ARE R-11, RELATION-11;
- SYNONYM IS RELATION-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.17 RESOURCE Section Header Statement

Purpose:

To allow a detailed description of the contents of a RESOURCE. A RESOURCE is something that is consumed by the target system. It is used in the target system to model system performance.

Syntax:

RESOURCE resource-name(s);

Usage Rules:

- It must be the first statement in a RESOURCE section.
- Several RESOURCES may be defined at once.

Synonyms:

RSC

Examples:

- RESOURCE CPU-TIME, MAN-POWER;
- RSC MONEY;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                   [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                                  { attv-name } [               { attv-name }  ]
      ATTRIBUTES ARE attr-name    {           } [ ,attr-name    {           }  ] .
                                  {  integer  } [               {  integer  }  ]
```

Complementary Statements:
none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CONSUMED Statement

Purpose:
To give the names of PROCESSORS that consume the RESOURCE.

Syntax:

```
        CONSUMED BY processor-name(s) AT RATE OF

                system-parameter PER resource-usage-parameter-name;
```

Complementary Statements:
CONSUMES statement in PROCESSOR section.

Usage Rules:
- More than one processor-name may be specified.

Synonyms:

CNSD

Examples:

- CONSUMED BY CPU AT A RATE OF 100,000 PER MINUTE;

- CNSD PROCESSOR-A, PROCESSOR-B RATE 9000 PER JOB;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
    THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
    ANY RELEVANT INFORMATION GOES HERE;
```

AD-A060 780 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78
F19628-76-C-0197

UNCLASSIFIED ESD-TR-78-130-VOL-2 NL

5 OF 6
AD
A060780

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

<u>KEYWORDS</u> ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS COM-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MEASURED Statement

Purpose:
To give the UNIT name that the RESOURCE is measured in.

Syntax:

```
MEASURED IN unit-name;
```

Complementary Statements:
MEASURES statement in UNIT section.

Usage Rules:
- A RESOURCE may be measured in only one UNIT.

Synonyms:

MSRD

Examples:

- MEASURED IN DOLLARS;

- MSRD MILLI-SECONDS;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:
RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

```
SECURITY IS security-name(S) ;
```

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-KENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
     SEE-MEMO memo-name(S) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

```
     SM          SEE-MEMOS
```

Examples:

- SEE-MEMO RW-05-23-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EDB-37, EDB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

SOURCE IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE R-11, RESOURCE-11;
- SYNONYM IS RESOURCE-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:
To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

```
TRACE-KEY trace-key-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:
- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.18 RESOURCE-USAGE-PARAMETER Section Header Statement

Purpose:
To allow a detailed description of RESOURCE-USAGE-PARAMETER(S).

Syntax:

RESOURCE-USAGE-PARAMETER resource-usage-parameter-name(s) ;

Usage Rules:
- Must be the first statement in a RESOURCE-USAGE-PARAMETER section.
- More than one RESOURCE-USAGE-PARAMETER may be defined at once.

Synonyms:

RUP

Examples:

- RESOURCE-USAGE-PARAMETER RUP-1;
- RUP DIFFICULTY-GRADING;

ASSERT Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                   [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
  coord-function arguments 2;

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                            { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name    {           } [ ,attr-name      {           } ] ...
                            {  integer  } [                 {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

- A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

KEYWORDS ARE keyword-name(s) ;

Complementary Statements:
APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

```
- KEYWORD IS PAYROLL;

- KEY IS CON-C1;

- KEYWORDS ARE EMP, EMPL, EMPLOYEE;
```

<u>RESOURCE-USAGE-PARAMETER-VALUE Statement</u>

Purpose:

To give the resource-usage-parameter-value (rup value) for the pair of RESOURCE-USAGE-PARAMETER and process.

Syntax:

```
     RESOURCE-USAGE-PARAMETER-VALUE :
                    system-parameter FOR process-name;
```

Complementary Statements:

RESOURCE-USAGE statement in PROCESS section.

Usage Rules:

- There may be at most one RESOURCE-USAGE-PARAMETER-VALUE for each unique pair of RESOURCE-USAGE-PARAMETER and PROCESS.

Synonyms:

RUP-VALUE	RUPV

Examples:

```
     - RESOURCE-USAGE-PARAMETER-VALUE:
               10 FOR PROCESS-1;

     - RUVP MAX-RATING PAYROLL-PROCESSING;
```

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:
To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:
RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;

- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:
To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN SYNONYM

Examples:

- SYNONYMS ARE R-11, RESOURCE-USAGE-PARAMETER-11;
- SYNONYM IS RESOURCE-USAGE-PARAMETER-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

<u>TRACE-KEY</u> trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

4.19 SET Section Header Statement

Purpose:

To allow a detailed description of a SET. For example, this section allows the PROBLEM-DEFINER to show how ENTITIES defined within the system are collected together for information processing purposes. SETS can be defined as physical or logical views of the data as seen by the user, designer, and/or programmer.

Syntax:

SET set-name(s) ;

Usage Rules:

-It must be the first statement in the SET section.

-Several SETS may be defined at a time.

Synonyms:

none.

Examples:

- SET FORECAST-INFO;

- SET TRANSACTION-INFO ;

ASSERT Statement

Purpose:

To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
     ASSERT name attribute-name attribute-value

                      [, name attribute-name attribute-value] ...;
```

Complementary Statements:

None.

Usage Rules:

- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                                { attv-name } [                 { attv-name } ]
ATTRIBUTES ARE attr-name        {           } [ ,attr-name      {           } ]..
                                {  integer  } [                 {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:
- It may be used in any section.
- A name may have several ATTRIBUTES

Synonyms:

ATTR	ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

CARDINALITY Statement

Purpose:
To define the number of times this SET appears in the system.

Syntax:

```
CARDINALITY IS system-parameter ;
```

Complementary Statements:
None.

Usage Rules:
-A SET may have only one CARDINALITY.

Synonyms:

CARD OCCS OCCURRENCES

Examples:

- CARDINALITY IS TEN;

- CARD FORTY-SEVEN;

CLASSIFICATION Statement

Purpose:

To associate security CLASSIFICATION requirements with data in the target system.

Syntax:

```
CLASSIFICATION classification-name [ integer ]
                    [, classification-name [ integer ]]... ;
```

Complementary Statements:

None.

Usage Rules:

- The name must be a CLASSIFICATION name.

Synonyms:

CLS   CLASSIFICATIONS

Examples:

- CLASSIFICATION IS PERSONNEL, SEC-LEVEL 3;
- CLS RING-LEVEL 2, UPDATE;

CONSISTS Statement

Purpose:

To describe the combination of INPUTS, OUTPUTS, and ENTITIES which make up this SET. This implies that each instance of the SET will contain values of the INPUT, OUTPUT and ENTITY names. An INPUT, OUTPUT or ENTITY may be repeated the number of times denoted by the SYSTEM-PARAMETER.

Syntax:

```
                                           input-
    CONSISTS OF [ system-parameter ]  output-name
                                           entity-

                                                input-
                 [ , [ system-parameter ]  output-name ] ... ;
                                                entity-
```

Complementary Statements:

CONTAINED statement in an ENTITY, INPUT or OUTPUT section.

Usage Rules:

- The names must be ENTITY, INPUT or OUTPUT names.
- A SET may contain several INPUTS, OUTPUTS, and ENTITIES.

Synonyms:

CSTS

Examples:

- CONSISTS OF DATA-ENTITY-1;
- CONSISTS OF: DATA-ENTITY-1, DATA-ENTITY-2;
- CSTS: ABSTRACT-1, ABSTRACT-2;

DERIVATION Statement

Purpose:
To express the specific system actions necessary to obtain the correct SET. This statement contains rules for DERIVATION which can be the DERIVED BY USING clause in the SET section.

Syntax:

```
DERIVATION ;
       comment-entry ;
```

Complementary Statements:
None.

Usage Rules:
- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DRVN

Examples:

```
- DERIVATION;
   THIS SET OF INFORMATION WAS DERIVED FROM THE PAYROLL FILES TO
THE OLD PAYSYSTEM;

DERIVATION;

   RULES FOR ADDITION:

ITEM MASTER-A ADDED WITH A TRANSACTION-CODE-74;
```

DERIVED Statement

Purpose:

To give a PROCESS that DERIVES values for the SET and the SETS, INPUTS, ENTITIES, GROUPS, and/or ELEMENTS used in the DERIVATION, and optionally, to specify conditions and/or iterations associated with the derivation.

Syntax:

```
                                      [         group-               ]
                                      [        entity-               ]
DERIVED BY  process-name(s) [ USING       set-      name(s) ]
                                      [         input-               ]
                                      [       element-               ]

      [DEPENDING ON element-    name(s) ]
      [             condition-          ]

      [             group-              ]
      [             entity-             ]
      [FOR EACH     element-    name(s) ];
      [             output-             ]
      [             input-              ]
      [             set-                ]
```

Complementary Statements:

DERIVES or USES statement in a PROCESS section and USED BY statement in a SET, INPUT, ENTITY, GROUP or ELEMENT section.

Usage Rules:

- Several PROCESSES may DERIVE values for a SET.

Synonyms:

DRVD USG DPNG DPG EC

Examples:

```
     - DERIVED BY PROCESS-A USING INPUT-1;

     - DERIVED BY PROCESS-1 USING ENTITY-A, ENTITY-B
             DPNG ON COND-A;

     - DRVD PROCESS-Q USG INPUT-1
             FOR EC GROUP-A, GROUP-B;

     - DRVD PROCESS-NAME USG ENTITY-A, GROUP-B
             DPNG ON COND-B
             FOR EC INPUT-A;
```

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
        comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:
To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

```
KEYWORDS ARE keyword-name(s) ;
```

Complementary Statements:
APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- It may be used in any section except the PROBLEM-DEFINER section.
- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

RESPONSIBLE-INTERFACE Statement

Purpose:
To give the INTERFACE which is responsible for this SET.

Syntax:

RESPONSIBLE-INTERFACE IS interface-name(s) ;

Complementary Statements:
RESPONSIBLE FOR in the INTERFACE section.

Usage Rules:

-The names must be INTERFACE names.

Synonyms:
RINT

Examples:

- RESPONSIBLE-INTERFACE IS PAYROLL-SYSTEM;
- RINT: ENGINEERING-DEPT;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section.
Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC	SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM	SEE-MEMOS

Examples:

```
- SEE-MEMO BW-05-03-75-01;

- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;

- SM EPB-37, EPB-38;
```

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

```
SOURCE IS source-name(s) ;
```

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;
- SOURCES ARE SDP-3-1,SDP-3-2,MEMO-23-MAY-1974;

SUBSET Statement

Purpose:

To show the structural relationship of this SET to higher-level SET(S). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

    SUBSET OF set-name(s) ;

Complementary Statements:

SUBSETS statement in SET section.

Usage Rules:

- The names in name(s) must be SET names.
- A SET may be a SUBSET of several other SETS.

Synonyms:

SST

Examples:

- SUBSET OF SET-GROUP-BANKS, SET-GROUP-CKTS;
- SST: STUDENT-INFO, COURSE-INFO;

SUBSETS Statement

Purpose:

To show the structural relationship of this SET to lower-level SET(S). This statement can be used to express a top-down or bottom-up view of the system.

Syntax:

SUBSETS ARE set-name(s) ;

Complementary Statements:

SUBSET statement in a SET section.

Usage Rules:

-The names must be SET names.

-Many SETS may be SUBSETS to one SET.

Synonyms:

SSTS

Examples:

- SUBSETS ARE SET-GROUP-BANKS, SET-GROUP-CKTS;

- SSTS: STUDENT-INFO, COURSE-INFO;

<u>SUBSETTING-CRITERIA Statement</u>

Purpose:
To indicate what data and/or rules are to be used to extract a portion of the data from the SET.

Syntax:

```
                                group-
SUBSETTING-CRITERIA ARE element-name(s) ;
                      subsetting-criterion-
```

Complementary Statements:
APPLIES statement in DEFINE section for SUBSETTING-CRITERION, and SUBSETTING-CRITERION statement in ELEMENT and GROUP sections.

Usage Rules:
-The names must be either ELEMENT or GROUP names.

-If the SUBSETTING-CRITERIA is an ELEMENT or a GROUP then it must be part of the ENTITY which is a legal member of this SET.

-A SET may have more than one SUBSETTING-CRITERIA.

-If a GROUP is given for the SUBSETTING-CRITERIA then the ELEMENTS which make up the GROUP taken together form the SUBSETTING-CRITERIA.

Synonyms:

SSCA

Examples:

- SUBSETTING-CRITERIA ARE GROUP-BANKS, GROUP-CKTS;

- SSCA: GROUP-107, GROUP-108;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- The statement may be used in any section except a MEMO section, or a DEFINE section for a SYNONYM.

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE S-11, SET-11;

- SYNONYM IS SET-11;

- SYN ALPHA;

TRACE-KEY Statement

Purpose:
To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

TRACE-KEY trace-key-name(s) ;

Complementary Statements:
APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:
- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;

- TKEY part-1, part-2;

UPDATED Statement

Purpose:

To indicate those PROCESSES which UPDATE this SET, and optionally, to specify the data used to do the UPDATING, or to specify conditions and/or iterations associated with the UPDATE.

Syntax:

```
                                     [           group-            ]
                                     [          entity-            ]
     UPDATED BY  process-name(s)     [ USING   element-    name(s) ]
                                     [           input-            ]
                                     [             set-            ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

Complementary Statements:

UPDATES or USES statement in PROCESS section and USED BY statement in INPUT, SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:

-A SET may be UPDATED by several different PROCESSES.

Synonyms:

UPDD USG DPNG DPG EC

Examples:

- UPDATED BY INPUT-PROCESS;

- UPDD PROC-1, PROC-2,PROC-789 DPNG ELE-A, ELE-B;

```
- UPDD PROC-3, DPNG ELE-C
      FOR EC INPUT-1, INPUT-2;
```

SET SECTION

USED Statement

Purpose:
    To indicate the PROCESS(ES) that USE(D) this SET, and optionally, DERIVE(S) OUTPUTS or UPDATE(S) SETS, ENTITIES, GROUPS, or ELEMENTS and to specify conditions and/or iterations associated with DERIVE(s) or UPDATE(s).

Syntax:

```
                                  [                  set-              ]
                                  [     { DERIVE } *output-            ]
     USED BY process-name(s) [ TO {        } entity-   name(s) ]
                                  [     { UPDATE }   group-            ]
                                  [                element-            ]

          [DEPENDING ON element-    name(s) ]
          [             condition-          ]

          [             group-              ]
          [             entity-             ]
          [FOR EACH     element-    name(s) ];
          [             output-             ]
          [             input-              ]
          [             set-                ]
```

* Output-name(s) may only be used with the DERIVE clause.

Complementary Statements:
    USES, UPDATES or DERIVES statement in a PROCESS section and DERIVED or UPDATED statement in SET, ENTITY, GROUP or ELEMENT sections.

Usage Rules:
    -Several PROCESSES may use a SET

Synonyms:

    DEV UED DPNG DPG FC

Examples:

```
     - USED BY PROCESS-INTEGER;

     - USED BY PROC-MU-A101, PROC-MU-A102 TO DERIVE OUTPUT-1
                 DPNG COND-A, COND-B
                  FOR EC INPUT-100;
```

VOLATILITY-MEMBER Statement

Purpose:
To give a measure of the changability of the contents of the SET.

Syntax:

```
VOLATILITY-MEMBER ;
        comment-entry ;
```

Complementary Statements:
None.

Usage Rules:
-Only one VOLATILITY-MEMBER statement may be given for any SET.

Synonyms:

VCIM

Examples:

```
- VOLATILITY-MEMBER;
     ALL THE ENTITIES ARE ACCESSED AT LEAST ONCE A WEEK;
```

VOLATILITY-SET Statement

Purpose:
To give a measure of the changability of the SET.

Syntax:

```
VOLATILITY-SET ;
        comment-entry ;
```

Complementary Statements:
None.

Usage Rules:
-Only one VOLATILITY-SET statement may be given for any SET.

Synonyms:

VOLS

Examples:

```
- VOLATILITY-SET;

    THIS SET WILL BE UPDATED TWICE DAILY ;
```

4.20 UNIT Section Header Statement

Purpose:

To allow a detailed description of a UNIT. A UNIT is something that is used in measuring a RESOURCE. It is used in recording and estimating the resource consumption in the target system.

Syntax:

<u>UNIT</u> name(s);

Usage Rules:

- It must be the first statement in a UNIT section.
- Several UNITS may be defined at once.

Synonyms:

None

Examples:

- UNIT MILLI-SECOND, DOLLAR;
- UNIT MAN-HOURS;

<u>ASSERT</u> Statement

Purpose:
To associate assertions about the attributes of names with other names for the purposes of consistency checking.

Syntax:

```
ASSERT name attribute-name attribute-value
                    [, name attribute-name attribute-value] ...;
```

Complementary Statements:
None.

Usage Rules:
- Name may be any type of name.

Synonyms:

ASRT

Examples:

```
- ASSERT data-name-1 type character;

- ASRT sine-function arguments 1,
       coord-function arguments 2;
```

ATTRIBUTES Statement

Purpose:
To specify properties or characteristics particular to a given section.

Syntax:

```
                              { attv-name } [              { attv-name } ]
ATTRIBUTES ARE attr-name      {           } [  ,attr-name  {           } ] .
                              {  integer  } [              {  integer  } ]
```

Complementary Statements:
none.

Usage Rules:

-A name may have several ATTRIBUTES

Synonyms:

ATTR ATTRIBUTE

Examples:

- ATTRIBUTES ARE FORMAT NUMERIC, LENGTH 6;
- ATTRIBUTES ARE FREQUENCY 100, VOLUME 10;
- ATTR CHAR ZZZ9V9;

DESCRIPTION Statement

Purpose:

To give a text DESCRIPTION of the section being described, and to state any information which cannot be easily or accurately stated with the syntax applicable for a given section.

Syntax:

```
DESCRIPTION ;
      comment-entry ;
```

Complementary Statements:

None.

Usage Rules:

- See chapter 2, section 10, for the rules concerning comment entries.

Synonyms:

DESC

Examples:

```
DESCRIPTION;
   THIS ALLOWS YOU TO DESCRIBE IN NARRATIVE FORM WHAT YOU EXPECT
THIS SECTION TO DO;

DESC;
   ANY RELEVANT INFORMATION GOES HERE;
```

KEYWORDS Statement

Purpose:

To selectively retrieve information from the URA data-base. A collection of information may be marked with a unique identifier (KEY) and later retrieved.

Syntax:

KEYWORDS ARE keyword-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for a keyword.

Usage Rules:

-A section may have several KEYWORDS

Synonyms:

KEY	KEYWORD

Examples:

- KEYWORD IS PAYROLL;
- KEY IS CON-C1;
- KEYWORDS ARE EMP, EMPL, EMPLOYEE;

MEASURES Statement

Purpose:
To give the RESOURCE names that the UNIT is used to measure.

Syntax:

<u>MEASURES</u> resource-name(s) ;

Complementary Statements:
MEASURED statement in RESOURCE section.

Usage Rules:
- A UNIT may measure several RESOURCES. A RESOURCE, however, may be measured only in one UNIT.

Synonyms:

MSRS

Examples:

- MEASURES CPU-TIME, REAL-TIME;

- MSRS FUNDS;

RESPONSIBLE-PROBLEM-DEFINER Statement

Purpose:

To associate the PROBLEM-DEFINER with those sections for which he is RESPONSIBLE.

Syntax:

RESPONSIBLE-PROBLEM-DEFINER IS problem-definer-name ;

Complementary Statements:

RESPONSIBLE FOR statement in PROBLEM-DEFINER section.

Usage Rules:

- Only one PROBLEM-DEFINER may be RESPONSIBLE for any section, hence, this statement may only be used once per section.

Synonyms:

RPD

Examples:

- RESPONSIBLE-PROBLEM-DEFINER IS AL-DICKEY;
- RPD A-HERSHEY;

SECURITY Statement

Purpose:

To associate SECURITY keys with a section which may be used to limit access to the information given in this section. Note: The SECURITY given refers to the Problem Statement information , not the information in the target system.

Syntax:

SECURITY IS security-name(s) ;

Complementary Statements:

APPLIES statement in a DEFINE section for a SECURITY.

Usage Rules:

- A name may have several SECURITIES.

Synonyms:

SEC SECURITIES

Examples:

- SECURITY IS PROJECT-MANAGER;
- SECURITIES ARE D-ORMISTON, S-MENNEL;
- SEC L-HANNON;

SEE-MEMO Statement

Purpose:

To indicate that information related to this section, and possibly other sections, is contained within the documentation. The information is contained in the MEMO(S) designated herein.

Syntax:

```
SEE-MEMO memo-name(s) ;
```

Complementary Statements:

APPLIES statement in a MEMO section.

Usage Rules:

- A section may have several such statements.

Synonyms:

SM   SEE-MEMOS

Examples:

- SEE-MEMO BW-05-03-75-01;
- SEE-MEMOS: PROJ-MGR-106, PROJ-MGR-109;
- SM EPB-37, EPB-38;

SOURCE Statement

Purpose:

To identify information not contained within the system documentation that is relevant to the understanding of the system. The SOURCE may be a person, a document (such as a practice or guideline), etc.

Syntax:

<u>SOURCE</u> IS source-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for SOURCE name.

Usage Rules:

- A name may have several SOURCES.

Synonyms:

SRC	SOURCES

Examples:

- SOURCE IS ENG-LETTER-1-MAY-1973;
- SOURCE: SDP-3-0;

SYNONYMS Statement

Purpose:

To give SYNONYMS for the name of the section. Can be used to define short forms for section-names in the documentation. Also can be used to resolve name conflicts within the system. Thus it is useful for reducing the manual effort of documentation.

Syntax:

SYNONYMS ARE synonym-name(s) ;

Complementary Statements:

DESIGNATE section.

Usage Rules:

- A name may have several SYNONYMS.

Synonyms:

SYN	SYNONYM

Examples:

- SYNONYMS ARE U-11, UNIT-11;
- SYNONYM IS UNIT-11;
- SYN ALPHA;

TRACE-KEY Statement

Purpose:

To associate a list of trace-keys with a name so that correspondences between objects in different data bases may be made.

Syntax:

<u>TRACE-KEY</u> trace-key-name(s) ;

Complementary Statements:

APPLIES statement in DEFINE section for TRACE-KEY name.

Usage Rules:

- The names in the name list must be trace-key names.

Synonyms:

TKEY

Examples:

- TRACE-KEY module-a;
- TKEY part-1, part-2;

APPENDIX A

Implementation Restrictions

A user-defined name can have a maximum length of 30 characters (letters, digits, dashes).

The User Requirements Analyzer (URA) will ignore card columns 73 through 80 (if card input is used). Thus, only columns 1 through 72 can be used for URL statements.

Each URL input line can contain either part of a URL statement or several statements.

Any URL statement may be broken anywhere a blank is allowed.

APPENDIX B

URL Reserved Words

A
AF
AFTER
AN
AND
APP
APPLIES
ARE
AS
ASOC
ASOD
ASRT
ASSERT
ASSOCIATED
ASSOCIATED-DATA
AT
ATTR
ATTRIBUTE
ATTRIBUTES
ATTRIBUTE-VALUE
ATTV
BEC
BECG
BECOMES
BECOMING
BECS
BEING
BETWEEN
BOX
BTWN
BY
CAL
CALLED
CARD
CARDINALITY
CAUSED
CAUSES
CLASSIFICATION
CLASSIFICATIONS
CLS
CNSS
CNSD
CNTD
COND
CONDITION
CONDITIONS
CONN
CONNECTIVITY
CONSISTS
CONSUMED

CONSUMES
CONTAINED
CSD
CSS
CSTS
DEF
DEFINE
DEPENDING
DEPENDS
DERIVATION
DERIVED
DERIVES
DESC
DESCRIPTION
DESG
DESIGNATE
DPG
DPND
DPNG
DPNS
DRV
DRVD
DRVN
DRVS
EACH
EC
ELE
ELEMENT
ELEMENTS
ENT
ENTITIES
ENTITY
EV
EVENT
EVENTS
EVERY
EVR
EVT
EVY
F
FALSE
FOR
FROM
GEND
GENERATED
GENERATES
GENS
GR
GROUP
GROUPS
HAP
HAPPENS
IDD

```
IDENTIFIED
IDENTIFIES
IDS
IN
INCC
INCEPTION
INCEPTION-CAUSES
INCP
INP
INPUT
INPUTS
INTD
INTERFACE
INTERFACES
INTERRUPTED
INTERRUPTS
INTERVAL
INTERVALS
INTF
INTS
IS
IT
KEY
KEYWORD
KEYWORDS
MADE
MAILBOX
MAILBOXES
MAINTAINED
MAINTAINS
MAK
MAKES
MBX
MEASURED
MEASURES
MEMO
MEMOS
MSRD
MSRS
MTND
MTNS
NEGINF
OCCS
OCCURRENCES
OF
ON
ORGANIZATIONAL-UNIT
ORGU
OUT
OUTPUT
OUTPUTS
PART
PD
```

PER
PERFORMED
PERFORMS
PFMD
PFMS
POSINF
PRC
PRCD
PRCR
PRD
PROBLEM-DEFINER
PROBLEM-DEFINERS
PROC
PROCEDURE
PROCESS
PROCESSES
PROCESSOR
PROCESSORS
PROCR
RCVD
RCVS
REAL-WOLRD-ENTITIES
REAL-WOLRD-ENTITY
RECEIVED
RECEIVES
REL
RELATED
RELATION
RELATIONS
RES
RESOURCE
RESOURCE-USAGE
RESOURCE-USAGE-PARAMETER
RESOURCE-USAGE-PARAMETER-VALUE
RESP
RESPONSIBLE
RESPONSIBLE-INTERFACE
RESPONSIBLE-PROBLEM-DEFINER
RINT
RLN
RPD
RPWE
RSC
RU
RUP
RUPV
RUP-VALUE
RWE
SEC
SECURITIES
SECURITY
SAR
SECURITY-ACCESS-RIGHT

SECURITY-ACCESS-RIGHTS
SEE-MEMO
SEE-MEMOS
SET
SETS
SM
SOURCE
SOURCES
SRC
SSCA
SSCN
SST
SSTS
SUBP
SUBPARTS
SUBSET
SUBSETS
SUBSETTING-CRITERIA
SUBSETTING-CRITERION
SYN
SYNONYM
SYNONYMS
SYSP
SYSPAR
SYSTEM-PARAMETER
SYSTEM-PARAMETERS
T
TERC
TERM
TERMINATED
TERMINATES
TERMINATION
TERMINATION-CAUSES
THE
THIS
THRU
THRU
TIMES-PER
TIMP
TKEY
TO
TRACE-KEY
TRGD
TRGS
TRIGGERED
TRIGGERS
TRMD
TRMS
TRUE
UNIT
UPDATE
UPDATED
UPD

UPDATES
UPDD
UPDS
USED
USES
USG
USING
UTLD
UTLS
UTILIZED
UTILIZES
VAL
VALUE
VALUES
VIA
VOL
VOLATILITY
VOLATILITY-MEMBER
VOLATILITY-SET
VOLM
VOLS
WHEN
WHETHER
WHILE
WHL
WT
WITH
WITHIN
WTH
WTN

APPENDIX C
UPL Optional Words

A
AN
AND
ARE
AS
AT
BEING
BY
FOR
FROM
IN
IS
IT
OF
ON
THE
THIS
TO
WHETHER
WITH

APPENDIX D

<u>Reserved Words with Synonyms</u>

AFTER	AF
APPLIES	APP
ASSERT	ASRT
ASSOCIATED	ASOC
ASSOCIATED-DATA	ASOD
ATTRIBUTE	ATTRIBUTES ATTR
ATTRIBUTE-VALUE	ATTV
BECOMES	BECS
BECOMING	BEC BECG
BETWEEN	BTWN
CALLED	CAL
CARDINALITY	CARD OCCS OCCURRENCES
CAUSED	CSD
CAUSES	CSS
CLASSIFICATION	CLASSIFICATIONS CLS
CONDITION	COND CONDITIONS
CONNECTIVITY	CONN
CONSISTS	CSTS
CONSUMED	CNSD
CONSUMES	CNSS
CONTAINED	CNTD
DEFINE	DEF
DEPENDING	DPGN DPG
DEPENDS	DPND DPNS
DERIVATION	DRVN
DERIVE	DRV
DERIVED	DRVD
DERIVES	DRVS
DESCRIPTION	DESC
DESIGNATE	DESG
EACH	EC
ELEMENT	ELE ELEMENTS
ENTITY	ENT ENTITIES
EVENT	EV EVT EVENTS
EVERY	EVR EVY
FALSE	F
GENERATED	GEND
GENERATES	GENS
GROUP	GR GROUPS
HAPPENS	HAP
IDENTIFIED	IDD
IDENTIFIES	IDS
INCEPTION	INCP
INCEPTION-CAUSES	INCC
INPUT	INP INPUTS
INTERFACE	INTF INTERFACES
	ORGANIZATIONAL-UNIT ORGU
	RWE REAL-WORLD-ENTITY

INTERRUPTED	INTD
INTERRUPTS	INTS
INTERVAL	INT INTERVALS
KEYWORD	KEY KEYWORDS
MADE	
MAILBOX	BOX MBX MAILBOXES
MAINTAINED	MTND
MAINTAINS	MTNS
MAKES	MAK
MEASURED	MSRD
MEASURES	MSRS
MEMO	MEMOS
NEGINF	
OUTPUT	OUT OUTPUTS
PART	
PER	
PERFORMED	PFMD
PERFORMS	PFMS
POSINF	
PROBLEM-DEFINER	PD PROBLEM-DEFINERS
PROCEDURE	PRCD PRD
PROCESS	PROC PRC PROCESSES
PROCESSOR	PRCR PROCR PROCESSORS
RECEIVED	RCVD
RECEIVES	RCVS
RELATED	REL
RELATION	RLN RELATIONS
RESOURCE	RSC
RESOURCE-USAGE	RU
RESOURCE-USAGE-PARAMETER	RUP
RESOURCE-USAGE-PARAMETER-VALUE	RUPV RUP-VALUE
RESPONSIBLE	RESP RES
RESPONSIBLE-INTERFACE	RINT
RESPONSIBLE-PROBLEM-DEFINER	RPD
SECURITY	SEC SECURITIES
SECURITY-ACCESS-RIGHT	SAR SECURITY-ACCESS-RIGHTS
SEE-MEMO	SM SEE-MEMOS
SET	SETS
SOURCE	SRC SOURCES
SUBPARTS	SUBP
SUBSET	SST
SUBSETS	SSTS
SUBSETTING-CRITERIA	SSCA
SUBSETTING-CRITERION	SSCN
SYNONYM	SYN SYNONYMS
SYSTEM-PARAMETER	SYSP SYSPAR SYSTEM-PARAMETERS
TERMINATED	TRMD
TERMINATES	TRMS
TERMINATION	TERM
TERMINATION-CAUSES	TERC
TIMES-PER	TIMP

TRACE-KEY	TKEY
TRIGGERED	TRGD
TRIGGERS	TRGS
TRUE	T
UNIT	
UPDATE	UPD
UPDATED	UPDD
UPDATES	UPDS
USED	
USES	
USING	USG
UTILIZED	UTLD
UTILIZES	UTLS
VALUES	VAL VALUE
VOLATILITY	VOL
VOLATILITY-MEMBER	VOLM
VOLATILITY-SET	VOLS
WHILE	WHL
WITHIN	WI WTN WTH

APPENDIX E

Name Types

ATTRIBUTE
ATTRIBUTE-VALUE
CLASSIFICATION
CONDITION
ELEMENT
ENTITY
EVENT
GROUP
INPUT
INTERFACE
INTERVAL
KEYWORD
MAILBOX
MEMO
OUTPUT
PROBLEM-DEFINER
PROCESS
PROCESSOR
RELATION
RESOURCE
RESOURCE-USAGE-PARAMETER
SECURITY
SOURCE
SET
SUBSETTING-CRITERION
SYNONYM
SYSTEM-PARAMETER
TRACE-KEY
UNDEFINED
UNIT

APPENDIX F

Section Types

CONDITION
DEFINE
DESIGNATE
ELEMENT
ENTITY
EVENT
GROUP
INPUT
INTERFACE
INTERVAL
MEMO
OUTPUT
PROBLEM-DEFINER
PROCESS
PROCESSOR
RELATION
RESOURCE
RESOURCE-USAGE-PARAMETER
SET
UNIT

APPENDIX G

URL Forms

The following hard-copy forms are intended to aid the user in writing URL according to the specifications given in the URL Reference Manual. The forms for a section give all statements allowed in that section and thus help the user to keep all possibilities in mind while writing his requirements. They also simplify the keypunching process.

CODING INSTRUCTIONS

The following general comments apply to the forms for all section types:

1. All statements are optional; the user should make use of only those he requires.

2. A continuation form is furnished for those statements which are too long for the space provided. To use this, the problem-definer should first state the section type and name at the top of the page, then, below, express the continuations as complete statements. (The abbreviations from Appendix D of the URL Reference Manual may be used for statement names.) A name-list should be broken only at the end of a name.

   DESIGNATE statements, of the form:

DESIGNATE name AS A synonym FOR name [, name AS A SYNONYM FOR name]...;

should be entered on continuation forms.

KEYPUNCHING INSTRUCTIONS

A statement should be keypunched only if it contains material coded by the user. For most statements, one may recognize the end of the statement by the semi-colon which is to be punched after it. The only exceptions to this rule are the comment-entry statements (DESCRIPTION, TRUE-WHILE, FALSE-WHILE, VOLATILITY, VOLATILITY-SET, VOLATILITY-MEMBER, DERIVATION, and PROCEDURE) which have two parts, each followed by a semi-colon. The first part consists of the printed statement name, while the second part contains only user-defined material. Both parts of a comment-entry statement should be keypunched if any coding appears in the second part of the statement. Otherwise, neither part of the statement should be punched.

Form titles, system name, dates and page numbers are not to be keypunched.

Columns 73-80 of each card will be ignored and therefore should not be used for URL statements. A URL statement may be punched on more than one card, and may be broken anywhere a blank is allowed.

URL CONDITION DEFINITION FORM

________________ system name   ________ date   PAGE ___ OF ___

CONDITION ____________________ ;
(condition name)

ASSERT ______________________________ ;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ________________ ________________ ,
(attribute name) (attribute value)

________________ ________________ ,

________________ ________________ ;

BECOMING TRUE CAUSES ____________________ ;
(list of event names)

BECOMING FALSE CAUSES ____________________ ;
(list of event names)

BECOMING TRUE INTERRUPTS ____________________ ;
(list of process names)

BECOMING FALSE INTERRUPTS ____________________ ;
(list of process names)

BECOMING TRUE TERMINATES ____________________ ;
(list of process names)

BECOMING FALSE TERMINATES ____________________ ;
(list of process names)

BECOMING TRUE TRIGGERS ____________________ ;
(list of process names)

BECOMING FALSE TRIGGERS ____________________ ;
(list of process names)

DEPENDS ON ______________________________ ;
(list of input, output, element, entity, group, set names)

DESCRIPTION;

____

____

____ ;
(narrative description)

URL CONDITION DEFINITION FORM

____________________ system name   __________ date   PAGE ___ OF ___

KEYWORDS ______________________________;
(list of keywords)

MADE TRUE BY ______________________________
(list of event, input, and process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

MADE FALSE BY ______________________________
(list of event, input, and process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

RESPONSIBLE-PROBLEM-DEFINER ____________________;
(name of responsible problem definer)

SECURITY ______________________________;
(list of applicable security names)

SEE-MEMO ______________________________;
(list of memo names)

SOURCE ______________________________;
(list of sources of information)

SYNONYMS ______________________________;
(list of synonyms)

TRACE-KEY ______________________________;
(list of trace-key names)

TRUE WHILE;

______________________________________;
(comment-entry)

FALSE WHILE;

______________________________________;
(comment-entry)

URL DEFINITION FORM

________________ system name	________ date	PAGE ___ OF ___

DEFINE ______________________________;
(name)

/_7 ATTRIBUTE;	/_7 SECURITY;
/_7 ATTRIBUTE-VALUE;	/_7 SOURCE;
/_7 CLASSIFICATION;	/_7 SUBSETTING-CRITERION;
/_7 KEYWORD;	/_7 SYSTEM-PARAMETER;
/_7 MAILBOX;	/_7 TRACE-KEY;

APPLIES TO ______________________________;
(list of appropriate names)
(only for keyword, mailbox, security, source and trace-key)

ASSERT ______________________________;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ________________ ________________,
(attribute name) (attribute value)

________________ ________________,

________________ ________________;

DESCRIPTION;

____

____

____

____

______________________________;
(narrative description)

KEYWORDS ______________________________;
(list of keywords)

URL DEFINITION FORM

____________ system name	________ date	PAGE ___ OF ___

MAINTAINED BY ________________________
(list of process names)
(only for subsetting-criterion)

DEPENDING ON ________________________
(list of element or condition names)

FOR EACH ________________________ ;
(list of group, entity, element, output, input, or set names)

RESPONSIBLE-PROBLEM-DEFINER ________________________ ;
(name of responsible problem definer)

SECURITY ________________________ ;
(list of applicable security names)

SEE-MEMO ________________________ ;
(list of memo names)

SOURCE ________________________ ;
(list of sources of information)

SUBSETTING-CRITERION FOR ________________________ ;
(list of set names)
(only for subsetting-criterion)

SYNONYMS ________________________ ;
(list of synonyms)

TRACE-KEY ________________________ ;
(list of trace-key names)

VALUE ________________________ ;
(value)
(only for system-parameter)

VALUES ________________________
(minimum value or NEGINF)
(only for system-parameter)
(may be used only if the VALUE statement is not used)

THRU ________________________ ;
(maximum value or POSINF
(only for system-parameter)
(may be used only if the VALUE statement is not used)

URL ELEMENT DEFINITION FORM

________________ system name   ________ date   PAGE ___ OF ___

ELEMENT ______________________;
(name of element)

ASSERT __________________________;
(list of names followed by attribute-names and attribute-values)

ASSOCIATED WITH ______________________;
(list of relation names)

ATTRIBUTES ARE ______________ ______________,
(attribute name)   (attribute value)

______________ ______________,

______________ ______________;

CLASSIFICATION ______________________;
(list of classification names optionally followed by classification levels)

CONTAINED IN ______________________;
(list of group, entity, input and output names)

DERIVED BY ______________________
(list of process names)

USING ______________________
(list of input, entity, set, group and element names)

DEPENDING ON ______________________
(list of element or condition names)

FOR EACH ______________________;
(list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

__________________________;
(narrative description)

IDENTIFIES ______________________;
(list of entity names)

KEYWORDS ______________________;
(list of keywords)

RESPONSIBLE-PROBLEM-DEFINER ________________;
(name of responsible problem definer)

URL ELEMENT DEFINITION FORM

________________ system name　________ date　PAGE ___ OF ___

SECURITY __________________________________;
(list of applicable security names)

SEE-MEMO __________________________________;
(list of memo names)

SOURCE __________________________________;
(list of sources of information)

SUBSETTING-CRITERION FOR ______________________;
(list of set names)

SYNONYMS __________________________________;
(list of synonyms)

TRACE-KEY __________________________________;
(list of trace-key names)

UPDATED BY __________________________________
(list of process names)

USING __________________________________
(list of input, set, entity, group and element names)

DEPENDING ON __________________________________
(list of element or condition names)

FOR EACH __________________________________;
(list of group, entity, element, output, input, or set names)

USED BY __________________________________
(list of process names)

TO DERIVE __________________________________
(list of set, entity, group, element and output names)

DEPENDING ON __________________________________
(list of element or condition names)

FOR EACH __________________________________;
(list of group, entity, element, output, input, or set names)

URL ELEMENT DEFINITION FORM

____________________ system name   __________ date   PAGE ___ OF ___

USED BY ______________________________
(list of process names)

TO UPDATE ______________________________
(list of set, entity, group and element names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

VALUE ______________________________;
(value)

VALUES ____________________ THRU ____________________;
(minimum value or NEGINF)   (maximum value or POSINF)
(may be used only if the VALUE statement is not used)

URL ENTITY DEFINITION FORM

______________________ system name ____________ date PAGE ___ OF ___

ENTITY ______________________; (name of entity)

ASSERT ______________________; (list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ______________ (attribute name) ______________ (attribute value),

______________ ______________;

CARDINALITY IS ______________________; (system-parameter)

CLASSIFICATION ______________________; (list of classification names optionally followed by classification levels)

CONSISTS OF ______________________; (list of group and element names, optionally preceded by system-parameters)

CONTAINED IN ______________________; (list of set names)

DERIVED BY ______________________ (list of process names)

USING ______________________ (list of input, set, entity, group and element names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________; (list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

______________________; (narrative description)

IDENTIFIED BY ______________________; (list of group and element names)

KEYWORDS ______________________; (list of keywords)

URL ENTITY DEFINITION FORM

PAGE ___ OF ___

______________________ system name ____________ date

RELATED TO ______________________ (entity name)

VIA ______________________ (relation name) ;

RESPONSIBLE-PROBLEM-DEFINER ______________________ (name of responsible problem definer) ;

SECURITY ______________________ (list of applicable security names) ;

SEE-MEMO ______________________ (list of memo names) ;

SOURCE ______________________ (list of sources of information) ;

SYNONYMS ______________________ (list of synonyms) ;

TRACE-KEY ______________________ (list of trace-key names) ;

UPDATED BY ______________________ (list of process names)

USING ______________________ (list of input, set, entity, group, or element names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________ ; (list of group, entity, element, output, input, or set names)

USED BY ______________________ (list of process names)

TO DERIVE ______________________ (list of set, entity, group, element, and output names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________ ; (list of group, entity, element, output, input, or set names)

USED BY ______________________ (list of process names)

TO UPDATE ______________________ (list of set, entity, group and element names)

URL ENTITY DEFINITION FORM

______________ system name	________ date	PAGE ___ OF ___

VOLATILITY;

____

____;
(comment-entry: changeability of the entity)

URL EVENT DEFINITION FORM

________________ system name   ________ date   PAGE ___ OF ___

EVENT ____________________ (name of event);

ASSERT ____________________ (list of names followed by attribute-names and attribute-values);

ATTRIBUTES ARE __________ (attribute name) __________ (attribute value),

__________ __________,

__________ __________;

CAUSED BY ____________________ (list of event and input names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

CAUSED WHEN ____________________ (name of condition) BECOMES TRUE;

CAUSED WHEN ____________________ (name of condition) BECOMES FALSE;

CAUSES ____________________ (list of event names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

____;
(narrative description)

HAPPENS __________ (system-parameter) TIMES-PER __________ (interval name);

URL EVENT DEFINITION FORM

________________ system name   ________ date   PAGE ___ OF ___

HAPPENS EVERY ____________ (system parameter) ____________ (interval name) ;

HAPPENS ____________ (system parameter) ____________ (interval name) AFTER ____________ (event) ;

HAPPENS WITHIN ____________ (system parameter) ____________ (interval name) AFTER ____________ (event) ;

ON INCEPTION OF ____________ (list of process names) ;

INTERRUPTS ____________ (list of process names)

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

KEYWORDS ____________ (list of keywords) ;

MAKES ____________ (list of condition names) TRUE

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

MAKES ____________ (list of condition names) FALSE

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

RESPONSIBLE-PROBLEM-DEFINER ____________ (name of responsible problem definer) ;

SECURITY ____________ (list of applicable security names) ;

URL EVENT DEFINITION FORM

PAGE ___ OF ___

____________________ system name   __________ date

SEE-MEMO ____________________________ ;
(list of memo names)

SOURCE ____________________________ ;
(list of sources of information)

SYNONYMS ____________________________ ;
(list of synonyms)

TERMINATES ____________________________
(list of process names)

DEPENDING ON ____________________________
(list of element or condition names)

FOR EACH ____________________________ ;
(list of group, entity, element, output, input, or set names)

ON TERMINATION OF ____________________________ ;
(list of process names)

TRACE-KEY ____________________________ ;
(list of trace-key names)

TRIGGERS ____________________________
(list of process names)

DEPENDING ON ____________________________
(list of element or condition names)

FOR EACH ____________________________ ;
(list of group, entity, element, output, input, or set names)

URL GROUP DEFINITION FORM

PAGE ___ OF ___

______________________ system name ______________ date

GROUP ______________________; (name of group)

ASSERT ______________________; (list of names followed by attribute-names and attribute-values)

ASSOCIATED WITH ______________________; (list of relation names)

ATTRIBUTES ARE ______________ (attribute name) ______________ (attribute value),

______________ ______________,

______________ ______________;

CLASSIFICATION ______________________; (list of classification names optionally followed by classification levels)

CONSISTS OF ______________________; (list of group and element names, optionally preceded by system-parameters)

CONTAINED IN ______________________; (list of group, entity, input and output names)

DERIVED BY ______________________ (list of process names)

USING ______________________ (list of input, entity, set, group or element names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________; (list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

______________________; (narrative description)

IDENTIFIES ______________________; (list of entity names)

KEYWORDS ______________________; (list of keywords)

URL GROUP DEFINITION FORM

PAGE ___ OF ___

_______________ system name   _______ date

RESPONSIBLE-PROBLEM-DEFINER _______________________;
(name of responsible problem definer)

SECURITY _______________________;
(list of applicable security names)

SEE-MEMO _______________________;
(list of memo names)

SOURCE _______________________;
(list of sources of information)

SUBSETTING-CRITERION FOR _______________________;
(list of set names)

SYNONYMS _______________________;
(list of synonyms)

TRACE-KEY _______________________;
(list of trace-key names)

UPDATED BY _______________________
(list of process names)

USING _______________________
(list of input, set, entity, group or element names)

DEPENDING ON _______________________
(list of element or condition names)

FOR EACH _______________________;
(list of group, entity, element, output, input, or set names)

USED BY _______________________
(list of process names)

TO DERIVE _______________________
(list of set, entity, group, element and output names)

DEPENDING ON _______________________
(list of element or condition names)

FOR EACH _______________________;
(list of group, entity, element, output, input, or set names)

URL GROUP DEFINITION FORM

PAGE ___ OF ___

______________________ system name

__________ date

USED BY ______________________________________
(list of process names)

TO UPDATE ______________________________________
(list of set, entity, group and element names)

DEPENDING ON ______________________________________
(list of element or condition names)

FOR EACH ______________________________________;
(list of group, entity, element, output, input, or set names)

URL INPUT DEFINITION FORM

____________________ system name   __________ date   PAGE ___ OF ___

INPUT ______________________________;
(name of input)

ASSERT ____;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ____________________ ____________________,
(attribute name)   (attribute value)

____________________ ____________________,

____________________ ____________________;

CAUSES ____
(list of event names)

DEPENDING ON ____
(list of element or condition names)

FOR EACH ____;
(list of group, entity, element, output, input, or set names)

CLASSIFICATION ____;
(list of classification names optionally followed by classification levels)

CONSISTS OF ____;
(list of group and element names, optionally preceded by system-parameters)

CONTAINED IN ____;
(list of set names)

DESCRIPTION;

____

____

____;
(narrative description)

GENERATED BY ____
(list of interface names)

DEPENDING ON ____
(list of element or condition names)

FOR EACH ____;
(list of group, entity, element, output, input, or set names)

URL INPUT DEFINITION FORM

____________ system name   ______ date   PAGE ___ OF ___

HAPPENS ____________ (system-parameter)

TIMES-PER ____________ (interval name) ;

HAPPENS EVERY ______ (system parameter) ______ (interval name) ;

HAPPENS ______ (system parameter) ______ (interval name) AFTER ______ (event) ;

HAPPENS WITHIN ______ (system parameter) ______ (interval name) AFTER ______ (event) ;

INTERRUPTS ____________ (list of process names)

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

KEYWORDS ____________ (list of keywords) ;

MAKES ____________ (list of condition names) TRUE

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

MAKES ____________ (list of condition names) FALSE

DEPENDING ON ____________ (list of element or condition names)

FOR EACH ____________ ;
(list of group, entity, element, output, input, or set names)

PART OF ____________ (name of input) ;

URL INPUT DEFINITION FORM

PAGE ___ OF ___

______________________ system name ____________ date

RECEIVED BY ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

RESPONSIBLE-PROBLEM-DEFINER ______________________________;
(name of responsible problem definer)

SECURITY ______________________________;
(list of applicable security names)

SEE-MEMO ______________________________;
(list of memo names)

SOURCE ______________________________;
(list of sources of information)

SUBPARTS ARE ______________________________;
(list of input names)

SYNONYMS ______________________________;
(list of synonyms)

TERMINATES ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

TRACE-KEY ______________________________;
(list of trace-key names)

TRIGGERS ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

URL INPUT DEFINITION FORM

_______________ system name   _______ date   PAGE ___ OF ___

USED BY ____________________ (list of process names)

TO DERIVE ____________________ (list of set, entity, group, element, and output names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________; (list of group, entity, element, output, input, or set names)

USED BY ____________________ (list of process names)

TO UPDATE ____________________ (list of set, entity, group, and element names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________; (list of group, entity, element, output, input, or set names)

URL INTERFACE DEFINITION FORM

PAGE ___ OF ___

________________ system name   ________ date

INTERFACE ______________________; (name of interface)

ASSERT ______________________; (list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE

____________ (attribute name)	____________, (attribute value)
____________	____________,
____________	____________,
____________	____________;

DESCRIPTION;

____

____

____

______________________; (narrative description)

GENERATES ______________________ (list of input names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________; (list of group, entity, element, output, input, or set names)

KEYWORDS ______________________; (list of keywords)

PART OF ______________________; (interface name)

RECEIVES ______________________ (list of output names)

DEPENDING ON ______________________ (list of element or condition names)

FOR EACH ______________________; (list of group, entity, element, output, input, or set names)

URL INTERFACE DEFINITION FORM

______ system name	______ date	PAGE ___ OF ___

RESPONSIBLE FOR ______________________________ ;
(list of set names)

RESPONSIBLE-PROBLEM-DEFINER ______________________________ ;
(name of responsible problem definer)

SECURITY ______________________________ ;
(list of applicable security names)

SECURITY-ACCESS-RIGHT ______________________________ ;
(list of classification names optionally followed by classification levels)

SEE-MEMO ______________________________ ;
(list of memo names)

SOURCE ______________________________ ;
(list of sources of information)

SUBPARTS ARE ______________________________ ;
(list of interface names)

SYNONYMS ______________________________ ;
(list of synonyms)

TRACE-KEY ______________________________ ;
(list of trace-key names)

AD-A060 780 MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERA--ETC F/G 9/2
USER REQUIREMENTS LANGUAGE (URL) USER'S MANUAL. PART II. (REFER--ETC(U)
JUL 78 F19628-76-C-0197
UNCLASSIFIED ESD-TR-78-130-VOL-2 NL

6 OF 6
AD
A060780

END
DATE
FILMED
01 -79
DDC

URL INTERVAL DEFINITION FORM

PAGE ___ OF ___

________________________ system name   ____________ date

INTERVAL ______________________________; (name of interval)

ASSERT ______________________________; (list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ____________________ ____________________,
(attribute name) (attribute value)

____________________ ____________________,

____________________ ____________________;

CONSISTS OF ______________________________; (list of interval names, optionally preceded by system-parameters)

DESCRIPTION;

____

____

____

______________________________; (narrative description)

KEYWORDS ______________________________; (list of keywords)

RESPONSIBLE-PROBLEM-DEFINER ______________________________; (name of responsible problem definer)

SECURITY ______________________________; (list of applicable security names)

SEE-MEMO ______________________________; (list of memo names)

SOURCE ______________________________; (list of sources of information)

SYNONYMS ______________________________; (list of synonyms)

TRACE-KEY ______________________________; (list of trace-key names)

URL MEMO DEFINITION FORM

PAGE ___ OF ___

________________________ system name ___________ date

MEMO ________________________ (memo name);

APPLIES TO ________________________ (list of section names);

ASSERT ________________________ (list of names followed by attribute-names and attribute-values);

ATTRIBUTES ARE ____________ (attribute name) ____________ (attribute value),

____________ ____________,

____________ ____________;

DESCRIPTION;

____

____

____

________________________ (narrative description);

KEYWORDS ________________________ (list of keywords);

RESPONSIBLE-PROBLEM-DEFINER ________________________ (name of responsible problem definer);

SECURITY ________________________ (list of applicable security names);

SOURCE ________________________ (list of sources of information);

SYNONYMS ________________________ (list of synonyms);

TRACE-KEY ________________________ (list of trace-key names);

URL OUTPUT DEFINITION FORM

PAGE ___ OF ___

______________________ system name   __________ date

OUTPUT ______________________________;
(name of output)

ASSERT ______________________________;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ______________ ______________,
(attribute name) (attribute value)

______________ ______________,

______________ ______________;

CLASSIFICATION ______________________________;
(list of classification names optionally followed by classification levels)

CONSISTS OF ______________________________;
(list of group and element names, optionally preceded by system-parameters)

CONTAINED IN ______________________________;
(list of set names)

DERIVED BY ______________________________
(list of process names)

USING ______________________________
(list of input, set, entity, group and element names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

______________________________;
(narrative description)

GENERATED BY ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

URL OUTPUT DEFINITION FORM

______________ system name ________ date PAGE ___ OF ___

HAPPENS ____________________ (system-parameter)

TIMES-PER ____________________ (interval name);

HAPPENS EVERY __________ (system parameter) __________ (interval name);

HAPPENS __________ (system parameter) __________ (interval name) AFTER ________ (event);

HAPPENS WITHIN __________ (system parameter) __________ (interval name) AFTER ________ (event);

KEYWORDS ____________________ (list of keywords);

PART OF ____________________ (name of output);

RECEIVED BY ____________________ (list of interface names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________ (list of group, entity, element, output, input, or set names);

RESPONSIBLE-PROBLEM-DEFINER ____________________ (name of responsible problem definer);

SECURITY ____________________ (list of applicable security names);

SEE-MEMO ____________________ (list of memo names);

SOURCE ____________________ (list of sources of information);

SUAPARTS ARE ____________________ (list of output names);

SYNONYMS ____________________ (list of synonyms);

TRACE-KEY ____________________ (list of trace-key names);

URL PROBLEM-DEFINER DEFINITION FORM

____________ system name	________ date	PAGE ___ OF ___

PROBLEM-DEFINER ______________________; (name of problem definer)

ASSERT ______________________________; (list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE ________________ ________________,
(attribute name) (attribute value)

________________ ________________,

________________ ________________;

DESCRIPTION;

____

____

______________________________;
(narrative description)

KEYWORDS ______________________________; (list of keywords)

MAILBOX ______________________________; (name of mailbox for problem definer)

RESPONSIBLE FOR ______________________________; (list of sections)

SECURITY IS ______________________________; (list of applicable security names)

SEE-MEMO ______________________________; (list of memo names)

SOURCE IS ______________________________; (list of sources of information)

SYNONYMS ______________________________; (list of synonyms)

TRACE-KEY ______________________________; (list of trace-key names)

URL PROCESS DEFINITION FORM

________________ system name   ________ date   PAGE __ OF __

PROCESS ____________________;
(name of process)

ASSERT ______________________________;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE _______________ _______________,
(attribute name)   (attribute value)

_______________ _______________,

_______________ _______________;

DERIVES ______________________________
(list of element, group, entity, set and output names)

USING ______________________________
(list of element, group, entity, set and input names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

______________________________;
(narrative description)

GENERATES ______________________________
(list of output names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

HAPPENS _______________ TIMES-PER _______________;
(system-parameter)   (interval name)

HAPPENS EVERY _______________ _______________;
(system parameter)   (interval name)

HAPPENS _______________ _______________ AFTER _______________;
(system parameter)   (interval name)   (event)

URL PROCESS DEFINITION FORM

PAGE ___ OF ___

_________________________ system name ____________ date

HAPPENS WITHIN ____________ (system parameter) ____________ (interval name) AFTER ________ (event);

INCEPTION-CAUSES ____________________ (list of event names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

INTERRUPTED BY ____________________ (list of event, input, and process names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

INTERRUPTED WHEN ____________________ (name of condition) BECOMES TRUE;

INTERRUPTED WHEN ____________________ (name of condition) BECOMES FALSE;

INTERRUPTS ____________________ (list of process names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

KEYWORDS ____________________ (list of keywords);

MAINTAINS ____________________ (list of relation or subsetting-criteria names)

DEPENDING ON ____________________ (list of element or condition names)

FOR EACH ____________________;
(list of group, entity, element, output, input, or set names)

URL PROCESS DEFINITION FORM

____________________ system name   __________ date   PAGE ___ OF ___

MAKES ______________________________ TRUE
(list of condition names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

MAKES ______________________________ FALSE
(list of condition names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

PART OF ______________________________;
(process name)

PERFORMED BY ______________________________;
(name of processor)

PROCEDURE;

____

____

____

____

____

____;
(comment entry: description of procedure)

RECEIVES ______________________________
(list of input names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

RESOURCE-USAGE __________________ FOR __________________;
(system-parameter)   (name of resource-usage-parameter)

URL PROCESS DEFINITION FORM

____________________	__________	PAGE ___ OF ___
system name	date	

RESPONSIBLE-PROBLEM-DEFINER ______________________________;
(name of responsible problem definer)

SECURITY ______________________________;
(list of applicable security names)

SECURITY-ACCESS-RIGHT ______________________________;
(list of classification names
optionally followed by
classification levels)

SEE-MEMO ______________________________;
(list of memo names)

SOURCE ______________________________;
(list of sources of information)

SUBPARTS ARE ______________________________;
(list of process names)

SYNONYMS ______________________________;
(list of synonyms)

TERMINATED BY ______________________________
(list of event, input, and process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

TERMINATED WHEN ____________________ BECOMES TRUE;
(name of condition)

TERMINATED WHEN ____________________ BECOMES FALSE;
(name of condition)

TERMINATES ______________________________
(list process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

URL PROCESS DEFINITION FORM

______________________ system name ____________ date PAGE ___ OF ___

TERMINATION-CAUSES ______________________________ (list of event names)

DEPENDING ON ______________________________ (list of element or condition names)

FOR EACH ______________________________; (list of group, entity, element, output, input, or set names)

TRACE-KEY ______________________________; (list of trace-key names)

TRIGGERED BY ______________________________ (list of event, input, and process names)

DEPENDING ON ______________________________ (list of element or condition names)

FOR EACH ______________________________; (list of group, entity, element, output, input, or set names)

TRIGGERED WHEN ______________________ (name of condition) BECOMES TRUE;

TRIGGERED WHEN ______________________ (name of condition) BECOMES FALSE;

TRIGGERS ______________________________ (list of process names)

DEPENDING ON ______________________________ (list of element or condition names)

FOR EACH ______________________________; (list of group, entity, element, output, input, or set names)

UPDATES ______________________________ (list of entity, set, group, and element names)

USING ______________________________ (list of input, set, entity, group, and element names)

DEPENDING ON ______________________________ (list of element or condition names)

FOR EACH ______________________________; (list of group, entity, element, output, input, or set names)

URL PROCESS DEFINITION FORM

_____________________ system name   ___________ date   PAGE ___ OF ___

USES ______________________________
(list of set, group, element, input, and entity names)

TO DERIVE ______________________________
(list of set, entity, group, element, and output names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

USES ______________________________
(list of set, group, element, input, and entity names)

TO UPDATE ______________________________
(list of set, entity, group, and element names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

UTILIZED BY ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

UTILIZES ______________________________
(list of process names)

DEPENDING ON ______________________________
(list of element or condition names)

FOR EACH ______________________________;
(list of group, entity, element, output, input, or set names)

URL PROCESSOR DEFINITION FORM

________________ system name   ________ date   PAGE ___ OF ___

PROCESSOR ____________________ (name of processor);

ASSERT ____________________ (list of names followed by attribute-names and attribute-values);

ATTRIBUTES ARE __________ (attribute name) __________ (attribute value),
__________ __________,
__________ __________;

CONSUMES ____________________ (name of resource) AT RATE OF

__________ (system-parameter) PER __________ (name of resource-usage-parameter);

DESCRIPTION;

____

____

____

____

______________________________ (narrative description);

KEYWORDS ____________________ (list of keywords);

PART OF ____________________ (processor name);

PERFORMS ____________________ (list of process names);

RESPONSIBLE-PROBLEM-DEFINER ____________________ (name of responsible problem definer);

SECURITY ____________________ (list of applicable security names);

URL PROCESSOR DEFINITION FORM

PAGE ___ OF ___

______________________ system name ___________ date

SECURITY-ACCESS-RIGHT ______________________________________; (list of classification names optionally followed by classification levels)

SEE-MEMO ______________________________________; (list of memo names)

SOURCE ______________________________________; (list of sources of information)

SUBPARTS ARE ______________________________________; (list of process names)

SYNONYMS ______________________________________; (list of synonyms)

TRACE-KEY ______________________________________; (list of trace-key names)

URL RELATION DEFINITION FORM

_______________ system name _______ date PAGE ___ OF ___

RELATION _______________; (name of relation)

ASSERT _______________; (list of names followed by attribute-names and attribute-values)

ASSOCIATED-DATA IS _______________; (list of element and group names)

ATTRIBUTES ARE _______ (attribute name) _______ (attribute value),

_______ _______,

_______ _______;

BETWEEN _______________ (name of entity)

AND _______________; (name of entity)

CARDINALITY IS _______________; (system-parameter)

CONNECTIVITY IS _______________ (system-parameter)

TO _______________; (system-parameter)

DERIVATION;

____

____

____

____

_______________; (derivation rules)

URL RELATION DEFINITION FORM

PAGE ___ OF ___

______________________ ____________
system name date

DESCRIPTION;

____

____

____

____

____;
(narrative description)

KEYWORDS ________________________________;
(list of keywords)

MAINTAINED BY ____________________________
(list of process names)

DEPENDING ON ____________________________
(list of element or condition names)

FOR EACH ________________________________;
(list of group, entity, element, output, input, or set names)

RESPONSIBLE-PROBLEM-DEFINER ______________;
(name of responsible problem definer)

SECURITY ________________________________;
(list of applicable security names)

SEE-MEMO ________________________________;
(list of memo names)

SOURCE __________________________________;
(list of sources of information)

SYNONYMS ________________________________;
(list of synonyms)

TRACE-KEY _______________________________;
(list of trace-key names)

URL RESOURCE DEFINITION FORM

______________ system name ______________ date PAGE ___ OF ___

RESOURCE ______________ (name of resource);

ASSERT ______________ (list of names followed by attribute-names and attribute-values);

ATTRIBUTES ARE ______________ (attribute name) ______________ (attribute value),

______________ ______________,

______________ ______________;

CONSUMED BY ______________ (list of processor names) AT RATE OF

______________ (system-parameter) PER ______________ (name of resource-usage-parameter);

DESCRIPTION;

____

____

______________ (narrative description);

KEYWORDS ______________ (list of keywords);

MEASURED IN ______________ (name of unit);

RESPONSIBLE-PROBLEM-DEFINER ______________ (name of responsible problem definer);

SECURITY ______________ (list of applicable security names);

SEE-MEMO ______________ (list of memo names);

SOURCE ______________ (list of sources of information);

SYNONYMS ______________ (list of synonyms);

TRACE-KEY ______________ (list of trace-key names);

URL RESOURCE-USAGE-PARAMETER DEFINITION FORM

____________________ system name   __________ date   PAGE ___ OF ___

RESOURCE-USAGE-PARAMETER ____________________;
(name of resource-usage-parameter)

ASSERT ____;
(list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE __________________ __________________,
(attribute name)   (attribute value)

__________________ __________________,

__________________ __________________;

DESCRIPTION;

____

____

____;
(narrative description)

KEYWORDS ______________________________________;
(list of keywords)

RESOURCE-USAGE-PARAMETER-VALUE ________________________
(system-parameter)

FOR ______________________________________;
(name of process)

RESPONSIBLE-PROBLEM-DEFINER ________________________;
(name of responsible problem definer)

SECURITY ______________________________________;
(list of applicable security names)

SEE-MEMO ______________________________________;
(list of memo names)

SOURCE ______________________________________;
(list of sources of information)

SYNONYMS ______________________________________;
(list of synonyms)

TRACE-KEY ______________________________________;
(list of trace-key names)

URL SET DEFINITION FORM

______________________ ____________ PAGE ___ OF ___
system name date

SET ____________________________________;
(name of set)

ASSERT ____________________________________;
(list of names followed by attribute-names
and attribute-values)

ATTRIBUTES ARE ____________________ ____________________,
(attribute name) (attribute value)

____________________ ____________________,

____________________ ____________________;

CARDINALITY IS ____________________________________;
(system-parameter)

CLASSIFICATION ____________________________________;
(list of classification names
optionally followed by classification levels)

CONSISTS OF ____________________________________;
(list of entity, input, and output names,
optionally preceded by system-parameters)

DERIVATION;

____

____;
(comment entry: derivation rules)

DERIVED BY ____________________________________
(list of process names)

USING ____________________________________
(list of input, set, entity, group and element names)

DEPENDING ON ____________________________________
(list of element or condition names)

FOR EACH ____________________________________;
(list of group, entity, element, output, input, or set names)

DESCRIPTION;

____

____

____;
(narrative description)

URL SET DEFINITION FORM

PAGE ___ OF ___

______________________ ______________
system name date

KEYWORDS ____;
(list of keywords)

RESPONSIBLE-INTERFACE ____________________________________;
(list of interface names)

RESPONSIBLE-PROBLEM-DEFINER ________________________________;
(name of responsible problem definer)

SECURITY ____;
(list of applicable security names)

SEE-MEMO ____;
(list of memo names)

SOURCE ____;
(list of sources of information)

SUBSET OF ____;
(list of set names)

SUBSETS ARE ____;
(list of set names)

SUBSETTING-CRITERIA ARE ______________________________;
(list of subsetting-criterion, element, and group names)

SYNONYMS ____;
(list of synonyms)

TRACE-KEY ____;
(list of trace-key names)

UPDATED BY ____
(list of process names)

USING ____
(list of input, set, entity, group, and element names)

DEPENDING ON ____
(list of element or condition names)

FOR EACH ____;
(list of group, entity, element, output, input, or set names)

URL SET DEFINITION FORM

PAGE ___ OF ___

_______________________ system name   ___________ date

USED BY ____
(list of process names)

TO DERIVE ____
(list of set, entity, group, element, and output names)

DEPENDING ON ____
(list of element or condition names)

FOR EACH ____;
(list of group, entity, element, output, input, or set names)

USED BY ____
(list of process names)

TO UPDATE ____
(list of set, entity, group, and element names)

DEPENDING ON ____
(list of element or condition names)

FOR EACH ____;
(list of group, entity, element, output, input, or set names)

VOLATILITY-MEMBER;

____

____;
(comment-entry: changeability of a member of the set)

VOLATILITY-SET;

____

____;
(comment-entry: changeability of the set)

URL UNIT DEFINITION FORM

_______________ system name   __________ date   PAGE ___ OF ___

UNIT ______________________; (name of unit)

ASSERT ______________________; (list of names followed by attribute-names and attribute-values)

ATTRIBUTES ARE __________ (attribute name) __________ (attribute value),

__________ __________,

__________ __________;

DESCRIPTION;

____

____

______________________; (narrative description)

KEYWORDS ______________________; (list of keywords)

MEASURES ______________________; (list of resource names)

RESPONSIBLE-PROBLEM-DEFINER ______________________; (name of responsible problem definer)

SECURITY ______________________; (list of applicable security names)

SEE-MEMO ______________________; (list of memo names)

SOURCE ______________________; (list of sources of information)

SYNONYMS ______________________; (list of synonyms)

TRACE-KEY ______________________; (list of trace-key names)

URL CONTINUATION FORM

____________	______	PAGE ___ OF ___
system name	date	

______________ ______________;
(section type) (name)